TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových
inženýrských studií

DIPLOMOVÁ PRÁCE

Třívrstvá aplikace pro párování nabídky a poptávky - část týkající se nalezení nejvýhodnější protinabídky

Three tier application for matching supplies and demands - part concerning locating of the most profitable return offer

Liberec 2005 Zbyněk Exner

Abstrakt:

Navržený systém je určen k internetovému obchodování formou aukce. Jedná se o třívrstvou aplikaci využívající technologie PHP, MySQL a server Apache. Práce se zabývá principy WWW programování včetně vzdálené správy dat.

Anotace:

Cílem diplomové práce je návrh a realizace systému pro internetové obchodování v rámci liberecké Agrární komory. Tento problém řeší projekt „Tržiště“, který je podporován firmou ITI Computers s.r.o. , jež spolupracuje s výše zmíněnou komorou. K základním požadavkům patří naprogramování internetové aplikace, která umožňuje evidenci členů Agrární komory a vytvoření systému pro obchodování formou aukce.

Náplní práce je uživatelská část systému, což znamená, že řeší přístup běžných uživatelů – kupujícího a prodávajícího. Součástí není administrátorský modul pro správu a konfiguraci systému.

K základním možnostem uživatelů patří vytváření nových aukcí (prodej) a přihazování na existující aukce (nákup). Dále je k dispozici, mimo jiného, vyhledávání aukcí podle několika kriterií, uživatelé si mohou nastavovat svůj profil (přístupové heslo, email apod.), mají možnost přehledně si zobrazovat veškeré své aukce a lze hodnotit ostatní uživatele z hlediska solidnosti. Systém je také vybaven automatickým rozesíláním informativních emailů pro uživatele, dojde-li ke změně stavu aukce.

Ke splnění výše zmíněných požadavků a problémů vedla realizace pomocí třívrstvé architektury. Využívá se aplikační server Apache, na kterém jsou spouštěny jednotlivé programy v PHP. K uložení dat slouží databázový server MySQL. Pro tvorbu samotných webových stránek je použit jazyk HTML a JavaScript. Softwarový projekt, zahrnující implementaci uživatelské části dynamické internetové aplikace, je součástí této diplomové práce.

Resume:

The aim of the diploma thesis was to design and realize an internet business system within Agrarian Chamber in Liberec. This task was solved by a project “Tržiště”, supported by ITI Computers s.r.o. company. It is a system that serves as an online auction for commodity dealings.

An application using three tier architecture consisting of PHP, MySQL and server Apache technologies was used. HTML and JavaScript languages were utilized to create the web pages.

The core of my work lies in the user part of the system. It deals with the interface for common users – buyers and sellers. It does not include the administrative part which is for management and configuration of the system.

Prohlášení:

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/200 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé DP a prohlašuji, že souhlasím s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum: 20. května 2005

Podpis:

Úvodem bych rád poděkoval panu RNDr. Ladislavu Mečířovi, vedoucímu mé diplomové práce, za podmětné připomínky, které vycházejí z jeho bohatých zkušeností s internetovými aplikacemi.

Dále patří poděkování konzultantovi panu Ing. Marianu Horňákovi ze společnosti ITI Computers s.r.o., s jehož pomocí jsme stanovili konkrétní požadavky a cíle práce.

Obsah:

1. Úvod

Posláním diplomové práce, zadané společností ITI Computers, je vytvořit určitý systém obchodování v rámci Agrární komory v Liberci.

V současné době existuje v tomto regionu mnoho zemědělských firem, které mezi sebou obchodují, ovšem jejich problémem je vzájemná komunikace. Každá firma, která nabízí určitý produkt, má zájem o to, prodat tento produkt za co nejvyšší cenu, tzn. uzavřít obchod s právnickou nebo fyzickou osobou, která nabídne nejvyšší cenu. Takto se dohodnout je pomocí běžných komunikačních prostředků, jako jsou telefony, faxy nebo emaily, velice obtížné a proto jsme, společně s firmou ITI Computers, která spolupracuje s libereckou Agrární komorou, navrhli projekt Tržiště, který tento komunikační proces velice usnadní.

Projekt Tržiště bude kromě samotného obchodování sloužit i k evidenci členů agrární komory, takže jeho součástí bude i databáze zahrnující informace o registrovaných členech.

Hlavní náplní je ovšem vlastní systém obchodování a to formou aukce. Stejně jako probíhají „normální“ aukce v aukčních síních, funguje i náš systém v elektronické podobě na webových stránkách.

Obecné požadavky zadavatele na náš systém vycházely z jeho povědomí o e-Bay systémech provozovaných v Americe nebo o systému Allegro provozovanému v Polsku. U nás rovněž existují podobné systémy, jako je např. Aukce.cz nebo Aukro.cz a další. Základní principy všech těchto systémů jsme pochopitelně implementovali i do našeho systému, protože tato obecná pravidla pro aukční obchodování se řídí standardními postupy a aplikace je proto musí respektovat.

Po analýze zmíněných systémů a po definování konkrétních a specifických požadavků zadavatele bylo zřejmé, že žádný z výše uvedených systémů není vyhovující a je tedy nutné navrhnout a vytvořit vlastní systém.

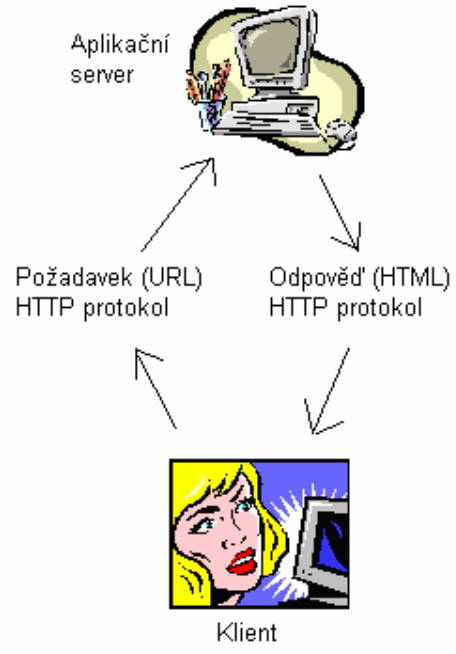
Hlavním přínosem této diplomové práce je tedy specifický systém, který je přesně „šitý“ na míru zadavateli. Jedná se o grafické rozvržení stránek, základní ovládání aplikace a o další speciální funkce, jako je např. automatické rozesílání emailů, vzájemné hodnocení uživatelů, registrování nových uživatelů, dělení kategorií a další.

Jak již název diplomové práce napovídá, tak společně se mnou pracuje na projektu i kolega Martin Tobolka, který vytváří tzv. administrátorskou část, ta bude podrobněji popsána v kapitole 3.5.

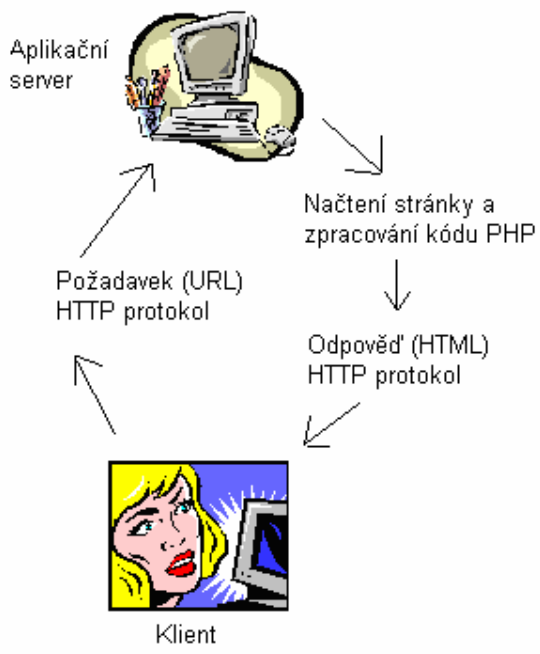
2. Teoretické základy práce

2.1. Aplikační servery

Při zjednodušeném pohledu na princip fungování internetu (obr. 1), je vidět, že při zadání internetové adresy na počítači (klient) se tato adresa, včetně názvu stránky, odešle do internetové sítě. Zde se podle adresy stránky najde počítač, na kterém je stránka uložena, a ten ji posléze pošle zpět formou odpovědi. Takovýto počítač se nazývá aplikační server (někdy se používá název webový) server. Takto jednoduše fungují stránky se statickým obsahem, ovšem dnes se častěji používají dynamické stránky, což znamená, že na aplikačním serveru běží nějaký program, který podle konkrétních uživatelských požadavků vytvoří stránku s vymezeným obsahem. Příklad takovéhoto zpracování pomocí jazyka PHP je na (obr. 2).



Obr.1 Statické stránky na internetu



Obr.2 dynamické stránky v kódu PHP

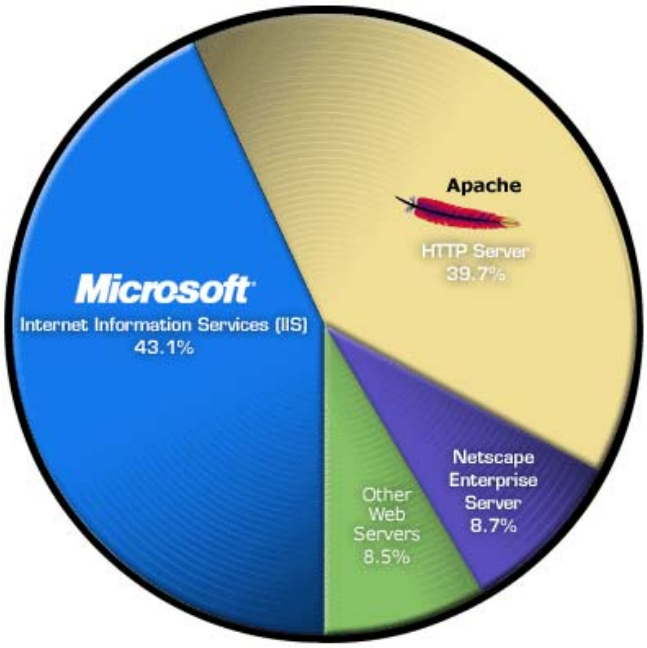
Aplikační server může být klasické PC (Personal Computer) používající operační systém MS Windows nebo Unix či velký sálový počítač pracující se systémem Unix.

V současnosti existuje na trhu v oblasti aplikačních serverů mnoho řešení, nicméně nejvýznamnějšími představiteli jsou **Microsoft IIS** (Internet Information Server) a **Apache**.

IIS je součástí operačních systémů Windows 2000 Professional a Windows XP Professional. Jedná se o velmi výkonný server, který vítězí na trhu v oblasti služeb pro velké společnosti a nejvíce navštěvované internetové servery (obr. 3). Jeho nevýhodou je špatná kompatibilita s ostatními operačními systémy, než jsou produkty společnosti Microsoft.

Naproti tomu Apache pracuje nejlépe v prostředí Unix, ale i pod Windows funguje bez problémů. Je to volně dostupný (open source) a rovněž vyspělý nástroj pro provoz webové prezentace, který je vzhledem ke své nulové ceně velice populární pro malé firmy a jednotlivce.

Co se týče dalších aplikačních serverů, tak stojí za zmínku ještě Netscape Enterprise Server od firmy Netscape Communications Corporation. Na trhu mu patří třetí místo. K jeho dobrým vlastnostem patří jednoduchá správa a široké spektrum platforem.



Obr. 3 Podíl webových serverů u 1000 nejnavštěvovanějších stránek
zdroj : studie společnosti Port80 software

2.2. Databázové systémy

2.2.1. Co je databáze?

Obecně si pod tímto pojmem lze představit soubor uložených dat, jež popisují nějakou oblast lidské činnosti, na základě kterých jsme schopni o této oblasti čerpat informace. Může se např. jednat o databázi odběratelů zboží z nějaké velkoprodejny a údaje o nich mohou sloužit k rozesílání nabídek na různé reklamní akce nebo k vyhodnocování jejich nákupů. Přitom na formě databáze nezáleží. Většinou sice přepokládáme, že jde o data uložená v počítači, ale ještě i dnes existuje mnoho databází v papírové formě uložených v různých pořadačích. Např. většina obvodních lékařů si stále údaje o pacientech zapisuje do karet a jediným nástrojem pro efektivní vyhledávání je jejich řazení podle abecedy. To samozřejmě velmi urychlí vyhledání určitého pacienta, ale při požadavku zjistit pacienty se stejnou chorobou, se bude muset lékař spolehnout na svojí paměť nebo projít karty všech pacientů.

Na databázi se tedy můžeme dívat jako na velkou skříň s informacemi a jde o to, aby práce s ní byla efektivní a umožňovala rychlé vyhledávání, filtrování a seskupování údajů podle požadavků uživatele. [3]

2.2.2. Vývoj databázových systémů

2.2.2.1 Manuální systémy (kartotéky)

Prvním pokusem o uchování informací v nějakém celku byli manuální systémy (kartotéky). Umožňovali jednoduše přidávat i mazat informace, jednotlivé záznamy šlo lépe vyhledat, ovšem složitější vyhledávání již nebylo možné (viz.před.kapitola). Jejich největší nevýhodou byly velké náklady na údržbu dat.

2.2.2.2 Agendové informační systémy

S příchodem prvních programovacích jazyků vznikly tzv. agendové informační systémy. Princip spočíval v tom, že pro každý problém existoval samostatný program, tzn. že data se tvořily pro program, izolovaně od ostatních vstupních dat pro jiné programy téhož informačního systému. Jestliže vznikla nějaká nová potřeba uživatele, bylo třeba vytvořit nový program. Největší nevýhodou byla závislost programu a dat. Důsledkem bylo zvýšení nákladů na údržbu a pořizování dat, tzv. redundace, a rovněž nekompatibilita, např.

významově stejná data se vyskytovala v rozdílně definovaném typu. Hrozilo tak porušení integrity dat, protože souvislosti mezi daty nebudou důvěryhodné [1].

2.2.2.3 Báze dat

Nedostatky agendového systému vyřešila báze dat. Jedná se o soustředění všech potřebných dat, se kterými programy pracují, do jednoho centrálního bloku. Tím se odstranila redundance a porušení integrity dat, ale hlavní nevýhoda - závislost dat na programu, stále přetrvávala [1].

2.2.2.4 Databázový systém

Databázový systém je tvořen dvěmi částmi. K bázi dat je přidán SŘBD (Systém Řízení Báze Dat), v anglické terminologii často označován jako DBMS (Data Base Managmenet System). SŘBD je program nebo soubor programů, který data organizuje. SŘBD dodávají výrobci programového vybavení podobně jako operační systémy, textové editory, nebo programy pro kreslení. Nejznámějšími produkty jsou např. *dBase, MS Access, Oracle, Informix, Progress, MS SQL, MySQL, Sybase* a další.

2.2.3. Úkoly SŘBD

SŘBD musí mít prostředky pro popis dat, k jejich definici a vytvoření vlastní databáze. Rovněž je třeba nějakého mechanismu , který spravuje data na disku a ví, kde je uložen konkrétní prvek. Tyto prostředky pro definici dat bývají označovány jako jazyk typu **DDL** (Data Definition Language) a slouží jako nástroj pro vytvoření a definici databáze.

Dále musí být k dispozici prostředky, které dovolí přistupovat k již existujícím databázím, vybírat z nich data, třídit je, filtrovat a měnit. Tyto prostředky pro manipulaci s daty bývají označovány jako jazyk typu **DML** (Data Manipulation Language).

Když chceme s databázovým systémem pracovat, předpokládáme, že je možné se spolehnout na to, že data jsou správná, to znamená, že data odpovídají vlastnostem příslušného popisovaného objektu reálného světa. Této vlastnosti se říká integrita. Systém musí zajistit, že integrita nebude porušena ani v případě havárie (např. výpadek proudu) ani zásahem jiného uživatele nebo aplikace, ať již

úmyslným nebo neúmyslným. Zajišťuje se jak na úrovni SŘBD, tak i zadáním určitých kriterií na úrovni definice dat, popřípadě na úrovni aplikační.

Shrneme-li tedy úkoly SŘBD, musí poskytovat následující služby: [3]

- Definice dat
- Údržba dat
- Manipulace s daty
- Zobrazení dat
- Zajištění integrity dat

SŘBD přináší také následující výhody: [2]

- Programátor nemusí znát organizaci ani fyzické uložení dat v bázi dat.
- Programátor přistupuje k datům pomocí jazyka strukturovaných dotazů SQL (z anglického Structured Query Language). (viz.kapitola 2.2.4.)

2.2.4. SQL

Jazyk SQL patří mezi tzv. dotazovací jazyky, což jsou prostředky umožňující formulovat požadavky uživatelů na výběr potřebných informací z databáze. Většinou se jedná o neprocedurální jazyky, to znamená, že příkazem se popisuje, co se má provést, jaká data potřebujeme, ale neurčuje se, jak se to má provést.

Již v 70. letech byl vyvinut firmou IBM dotazovací jazyk SQL (Structured Query Language), určený pro komunikaci se SŘBD relačního typu (viz. kapitola 2.2.5.4.). Vzhledem k tomu, že neposkytuje prostředky pro manipulaci s obrazovkou a pro uživatelský vstup a výstup, je spíš jakýmsi podjazykem. Jeho prostřednictvím lze interaktivně formulovat dotazy, nebo může být použit jako součást hostitelského jazyka pro přístup k datům, zatímco zbývající část aplikace je napsaná v jiném programovacím jazyce. SQL je tvořen v porovnání s ostatními jazyky pouze několika příkazy, zato však velmi výkonnými. S jeho pomocí lze definovat data (definovat strukturu tabulky), aktualizovat data (přidávat a mazat řádky a sloupce, vkládat data z vnějšího souboru), ale hlavní síla je v příkazu *SELECT* pro formulaci dotazů. Obliba a rozšíření jazyka SQL vedlo k jeho

standardizaci, nicméně implementace různých producentů SŘBD se od tohoto standardu vždy poněkud liší. [3]

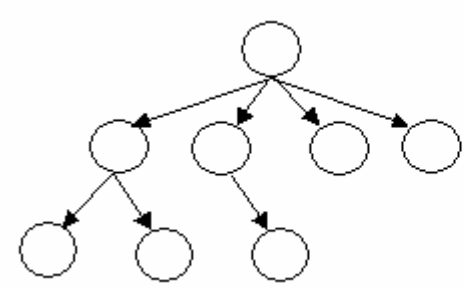
2.2.5. Implementační datové modely

Datový model popisuje strukturu databáze na konceptuální, tedy logické úrovni. Konceptuální schéma, které na této úrovni vzniká, je nezávislé na konkrétních implementačních nástrojích, tedy programech (SŘBD), které definované funkce a procesy realizují vlastními prostředky.

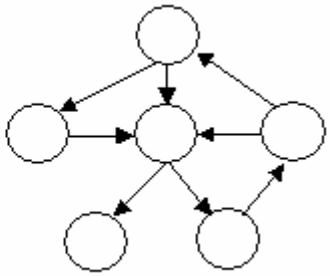
V průběhu historie vzniklo několik datových modelů, z nichž jako první byl **hierarchický** model dat (obr. 4), který vycházel z reálného světa - organizace moci, rozklad výrobků na součástky, strom adresářů apod. Pro hierarchické modelování je typická práce se stromy, kdy ve stromu jsou realizovány vztahy 1:N.

Variací hierarchického modelu je **síťový** model databáze (obr. 5). V síťovém modelování je možné vyjadřovat vedle vztahů 1:N i vztahy M:N. Fyzická realizace síťového modelu je náročná a aktualizace obvykle komplikovaná.

Základní výhodou hierarchického a síťového modelu je efektivnost zpracování, tj. rychlost přístupu k datovým záznamům. Na druhé straně mezi nevýhody hierarchických databází patří to, že je nesnadné jednou nadefinované stromy a vazby mezi nimi měnit. Nejsou uzpůsobeny pro dotazy.



obr. 4 Hierarchický model



obr. 5 Síťový model

2.2.5.1. Relační model

Koncepci relačního modelu poprvé formuloval a publikoval zaměstnanec IBM Ted Codd v roce 1970 jako odpověď na nedostatky tehdejších databázových řešení založených na síťovém a hierarchickém datovém modelu. Základní myšlenkou nového modelu bylo využití relačního kalkulu a algebry pro ukládání a manipulaci s daty. Codd prokázal, že pomocí základních operací (sjednocení,

kartézský součin, rozdíl, selekce, projekce a spojení) s relacemi lze uskutečnit veškeré základní operace s daty, a že ostatní operace jsou již jen kombinacemi těchto pěti základních operací.

Již první návrh relačního modelu předpokládal uložení dat v tabulkách, nezávislost dat na použitém hardwaru a způsobu fyzického uložení, a přístupu k datům pomocí neprocedurálního jazyka koncipovaného pro netechnické uživatele. Cílem bylo dát uživateli možnost specifikovat operaci nad jím definovanou množinou dat, místo procedurálního popisu jednotlivých kroků pro zpracování dat založeného na manipulaci s jedním záznamem. V průběhu sedmdesátých let byly implementovány první relační databáze (Ingres v UC-Berkeley - předchůdce nynější PostgreSQL, a Systém-R v IBM - předchůdce DB2), které v praxi ověřily životaschopnost a možnosti relačního modelu. Osmdesátá léta pak představovala počátek zlatého věku relačních databází (vznik Oracle, Sybase, InterBase, Informix).

Relační databáze je databáze sestavená z řady tabulek, jejichž sloupce jsou vázány na sloupce v jiných tabulkách. Takto propojená datová pole jsou na sobě určitým způsobem závislá. Jejich vztahy jsou založeny na klíčových hodnotách uložených v příslušných sloupcích.

U relačních databází je základní výhodou relativně snadná modifikace a propojování tabulek a s nimi spojená možnost dotazů. Slabým článkem je nízká efektivnost zpracování, což se projevuje v tom, že řada příkazů vyžaduje velké množství přístupů na disk a tím se zpomaluje zpracování.

2.2.5.2. Objektově orientovaný model dat

Principy objektově orientovaného programování (OOP) pronikly i do oblasti databází. Objektový přístup nabízí možnost lépe popsat objekty reálného světa než relační model, který popisuje pouze vlastnosti objektů, tedy statistická data. Na rozdíl od tabulek v relačním modelu mohou mít objekty kromě vlastností i metody, které představují funkce a procedury, které s nimi manipulují. Na základě společných rysů, lze vytvářet třídy objektů, které své vlastnosti a metody mohou dědit. Zdá se, že objektově orientované principy jsou pro modelování reality ideální, nicméně teoreticky objektově orientovaný model dosud není dostatečně popsán a standardizován, což je důvodem pomalejšího pronikání OOSŘBD do praxe. Spíše se projevují trendy rozšířit relační model o prvky a principy OOP,

takže můžeme mluvit o objektově relační technologii (ORSŘBD). Tento model je v současné době využíván například v databázovém stroji PostgreSQL a lze jej právem považovat za velmi nadějný model, který bude do budoucna hojně využíván.[1]

2.3. Statické internetové aplikace

Internetové aplikace (často označované jako stránky), lze z hlediska funkčnosti rozdělit na:

- *Statické*, jejichž obsah se nemění v závislosti na požadavcích uživatele a mají pouze informační charakter.
- *Dynamické*, které reagují na požadavky uživatele a podle jeho zadaní mění svůj vzhled a obsah. [2]

2.3.1. HTML

Zkratka HTML znamená *Hypertext Markup Language.* Jedná se o jazyk pro tvorbu dokumentů, který definuje vzhled textu (velikost nadpisů, použité fonty písma, okraje, ...). Jazyk HTML byl speciálně vyvinut (a stále se vyvíjí) za účelem publikování dokumentů na World Wide Webu. Původně vznikl ze značkovacího jazyka SGML (Standard Generalized Markup Language).

Jak je již z názvu patrné, původně byl jazyk HTML vytvořen pro zobrazování textové dokumentace. Avšak během relativně krátké doby byla jeho původní funkce doplněna o další multimediální prvky (grafika, animace, hudba). Tím se znásobila jeho hodnota, coby média pro přenos informace.

Zdrojový kód dokumentu psaném v jazyce HTML je prostý text psaný v ASCII formátu, který lze prohlížet i upravovat v libovolném textovém editoru. Jazyk HTML je jazykem typografickým, což znamená, že výsledný dokument pouze popisuje, ale jeho interpretace je přenechána až na cílový prohlížeč HTML dokumentů - např. Internet Explorer, Mozilla Firefox nebo Netscape Navigator.

Příkazy tohoto jazyka (značky) jsou spolu se svými parametry uzavírány do špičatých závorek <>. Každý takto napsaný příkaz nějakým způsobem definuje formátování elementů na stránce.

Značky jsou buď párové nebo nepárové. Párové jsou ty, které mezi sebe uzavírají vlastní obsah a přiřazují mu tak danou hodnotu, např. text . Nepárové pak umísťují do stránky jediný element, který je už svou

podstatou nedělitelný. Jdou to například elementy jako je pozadí, vodorovná linka či vložení obrázku: <img src=”obrazek.jpg”>.

Hlavní výhodou jazyka HTML je jeho jednoduchost. Standard jazyka stanovený konsorciem W3C (World Wide Web Consorcium), je ale velmi benevolentní k chybám ve značkování. Z toho vyplývají problémy s přenositelností stránek, protože každý prohlížeč má jinou metodu na odstraňování chyb ve značkování a výsledkem jsou tudíž rozdílné interpretace. [2]

2.3.2. XML a XHTML

V současné době se o XML (eXtensible Markup Language) hovoří nejčastěji v souvislosti s Webem a považuje se za nástupce dnes používaného jazyka HTML. Výhoda XML spočívá v tom, že autor stránky může používat vlastní značky, které dokáží mnohem přesněji označit význam prezentovaných informací.

Situace se dnes vyvíjí tak, že nám XML umožní rozšiřovat množinu elementů, které nám při tvorbě stránek nabízí HTML. Všechny značky jsou definovány buď přímo v dokumentu nebo pomocí externí deklarace v souboru DTD (Dokument Type Definition).

XML má daleko přísnější kontrolu syntaxe, takže chyby, které by v jazyce HTML byly připuštěny, jsou zde neakceptovány a interpretace dokumentu neproběhne. Tím pádem jsou stránky v XML snazší pro čtení než ty současné v HTML, které obsahují mnoho chyb. To umožňuje vývoj nových jednoduchých prohlížečů určených zejména pro různá kapesní mobilní zařízení.

Nevýhodou XML je malá kompatibilita se stávajícími prohlížeči internetových stránek. Z toho důvodu vznikl jazyk XHTML (eXtensible HyperText Markup Language), který vychází z jazyka HTML, ale řídí se principy jazyka XML. XHTML je zpětně kompatibilní se staršími prohlížeči, ale má pevně stanovenou syntaxi, která je kontrolována ještě před vlastním zpracováním stránky. Lze jej tedy použít i na vytvoření stránek, které budou zobrazitelné i na jiných periferiích než jsou stolní počítače. [2]

2.3.3. CSS

Kaskádové styly - CSS (Cascading Style Sheet) nabízí tvůrcům stránek daleko širší formátovací možnosti, a to nejen textu. Definice stylů je umístěna na

začátku stránky a jednotlivé značky se na ně v průběhu zobrazení a práce se stránkou odkazují.

Příklad zápisu:

```
<STYLE TYPE="text/css">
<!--
H2 {font-family: Verdana;
font-size: 16pt;
font-weight: bold;
color: blue;}
-->
</STYLE>
```

Tato definice existujícímu stylu *H2* (nadpisu druhé úrovně) přiřadí modrou barvu, velikost 16 bodů, písmo Verdana a celý nadpis bude vypsán tučně.

Velkou výhodou CSS je nadefinování způsobu zobrazení pro jednotlivé prvky a následně používání pouze těchto předdefinovaných stylů, což usnadňuje práci a zlepšuje přehlednost výsledného kódu.[2]

Poznámka:

Slovo kaskádové v názvu znamená, že jediný text může ovlivňovat několik stylů, které jej formátují. Nenastává totiž chyba, která by vyplývala z konfliktu dvou vlastností, které mají být ovlivněny dvěma různými styly, protože každý styl má jinou prioritu. U konfliktních stylů, které následují v definici stránky, má přednost ta definice, která byla uvedena jako poslední.

2.4. Dynamické internetové aplikace

Dynamické (interaktivní) internetové stránky reagují na požadavky uživatele a podle jeho chování mění svůj obsah.

Dynamické technologie lze rozdělit na technologie na straně klienta (Dynamické HTML, JavaScript, Java-aplety) a na straně serveru (CGI-skripty, PHP, Active Server Pages, Server Side JavaScript). Oby typy technologií lze pro větší účelnost navzájem kombinovat, např. PHP s klientským JavaScriptem při kontrole údajů v odesílaném formuláři.

2.4.1. CGI

CGI (Common Gateway Interface) je standard pro komunikaci externích aplikací s informačními servery, jako jsou např. servery WWW. Běžné dokumenty HTML poskytované serverem WWW jsou statické a proto mohou jen stěží odrážet často se měnící informace nebo zobrazovat údaje podle požadavku uživatele. Nastávající problém lze velmi snadno vyřešit právě pomocí programů podle standardu CGI.

CGI je implementováno jako součást serveru WWW. Umožňuje mu spouštět libovolné, na daném systému spustitelné programy a předávat jim parametry. Spuštěný program pak vstupní data zpracuje a výsledky předá prostřednictvím serveru WWW zpět prohlížeči WWW.

Struktura všech programů CGI je velmi podobná a odpovídá následujícímu schématu:

1. Načtení a analýza vstupních parametrů.
2. Vlastní činnost programu.
3. Generování výstupu programu. [4]

Program CGI může být napsán v jakékoliv jazyku, např. Perl, C, C++, příkazové shelly Unixu aj.

2.4.2. PHP

Historie PHP (Personal Home Page) sahá do roku 1994, kdy Rasmus Lerdorf vytvořil první skripty pro evidování přístupu k jeho stránkám. Později k tomuto systému přidal i nástroj, který umožňoval začleňování SQL dotazů do stránek, vytváření formulářů a zobrazování výsledků dotazů.

Dnes je PHP vyvíjeno jako Open Source (volně šiřitelný software s otevřeným kódem) a jeho poslední oficiální verze je 5. PHP (dnes je uváděna zkratka PHP Hypertext Preprocesor) je skriptovací jazyk zabudovaný na straně serveru s velmi podobnou syntaxí jakou má jazyk C/C++.

PHP pracuje uvnitř dokumentu HTML a podle proměnných, které reprezentují požadavky klienta, je tento dokument zpracován a jako výsledek je klientovi zaslána odpověď ve formátu dokumentu HTML. Kód jazyka PHP je do HTML dokumentu vložen pomocí speciálních značek, začátek uvozují „<?php“, pak následuje vlastní kód a konec je označen pomocí „?>“. Lze použít i zkráceného zápisu „<?, kód, ?>“. [2]

Velkou výhodou PHP je jeho univerzálnost a kompatibilita s různými platformami jako jsou MS Windows, Unix či Linux a různými aplikačními servery jako třeba Apache nebo IIS.[2]

2.4.3. ASP

Spolu s PHP je ASP (Active Server Pages) od společnosti Microsoft nejvíce rozšířeným skriptovacím strojem innternetu. Jedná se rovněž o skriptovací jazyk na straně serveru, který je vložen do stránky HTML a podle předaných proměnných od klienta je dokument zpracován a odeslán zpět klientovi. Syntaxe ASP je jiná než u PHP, do značné míry je podobná MS Visual Basic, do dokumentu je vložena pomocí značek „<% %>“.

Nevýhodou oproti PHP je fakt, že se jedná o komerční produkt, nikterak levný, a realizace je svázána s použitím WWW serveru od firmy Microsoft.

2.4.4. Java

Java je programovací jazyk vyvinutý firmou SUN Microsystems. Je velmi podobný jazyku C++, ovšem zbavený některých rysů, které působily problémy a byly zdrojem velmi častých chyb.

Pomocí tohoto jazyka lze vytvářet stránky JSP (Java Server Pages), které jsou na straně serveru zpracovány JavaServletem. JavaServlet je program, který obsluhuje požadavky klienta ve formě http a výsledky vrací klientovi ve stejné formě. Princip vložení kódu do dokumentu HTML je opět totožný jako u ASP a PHP [2].

Velkou výhodou Javy je její nezávislost na platformě, naopak jako relativní nevýhodu lze označit poměrně velkou složitost tohoto jazyka, která často způsobuje problémy začínajícím programátorům.

2.4.5. JavaScript a Server-Side JavaScript

JavaScript (na straně klienta) je jednoduchý jazyk se syntaxí vycházející z jazyka Java. S JavaScriptem přišla firma Netscape Communications Corporation a zabudovala jej do svého prohlížeče Netscape Navigator, dnes je ovšem JavaScript podporován všemi používanými prohlížeči.

JavaScript se zapisuje přímo do HTML kódu a lze ho použít v mnoha situacích. Jeho nejčastější použití je ve spojení s formuláři. Krátké skripty

v JavaScriptu mohou kontrolovat správnost údajů v polích formuláře ještě před odesláním na server. Uživatel tak získá nesrovnatelně rychlejší odezvu v porovnání s klasickým způsobem využívajícím pouze CGI-skripty. Další častou oblastí použití JavaScriptu jsou drobná vylepšení interaktivnosti stránek – celkem snadno lze např. zařídit, aby odkaz měnil barvu po přejetí myši.

Úspěch JavaScriptu byl tak obrovský, že se firma Netscape Communications Corporation rozhodla pro využití JavaScriptu na straně serveru – tzv. SSJS (Server-Side JavaScript). Princip je opět totožný s výše jmenovanými jazyky, tedy kód je do stránky HTML vložen pomocí značek (tentokrát „<SERVER> a </SERVER>“) a je pak zpracován na straně serveru.

Výhodou SSJS je to, že používá jednoduchý a oblíbený jazyk JavaScript, naopak nedostatkem je závislost na aplikačním serveru Enterprise Server od společnosti Netscape Communications Corporation, který svými parametry značně zaostává za MS IIS i Apache.

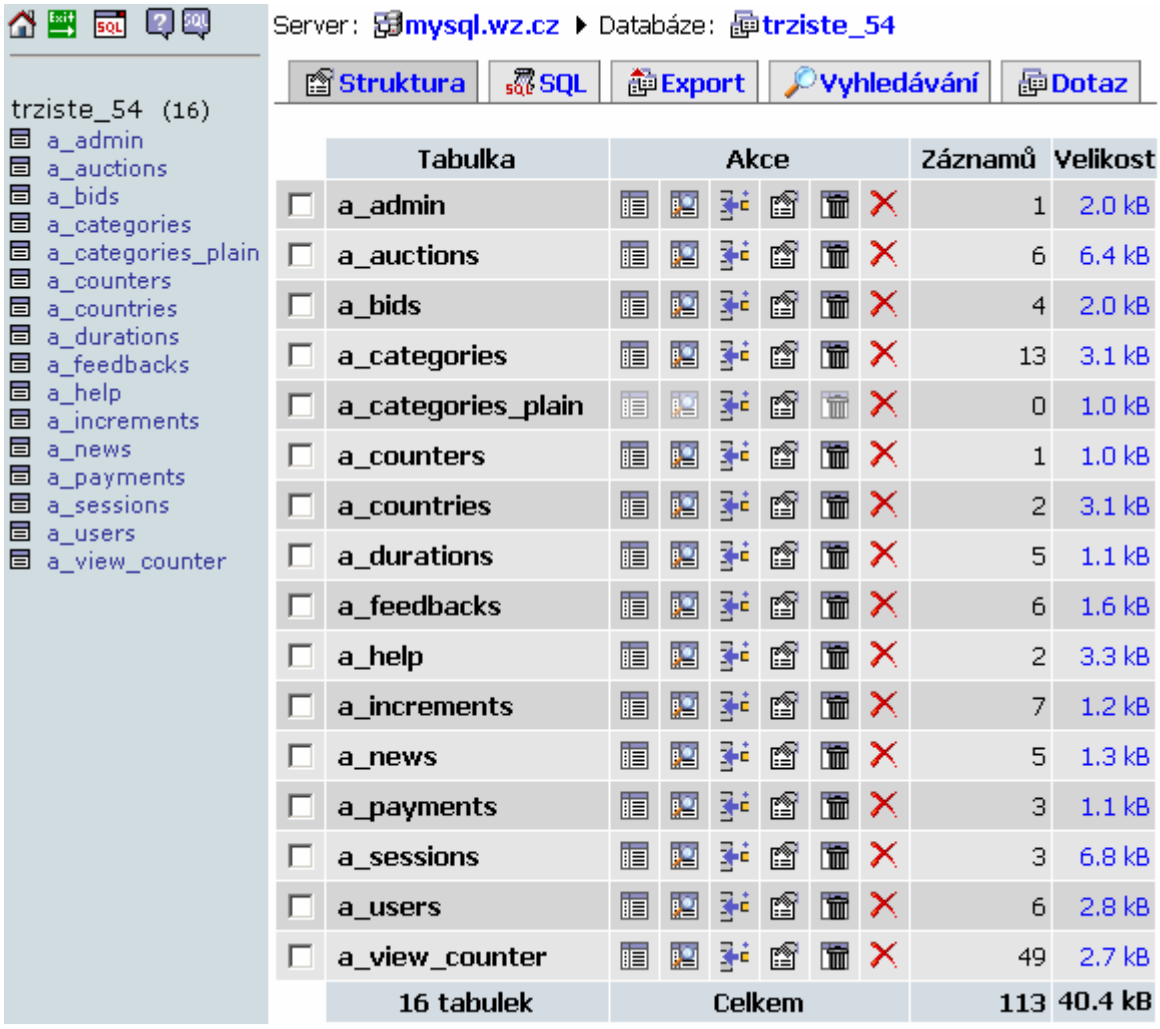
2.5. Nástroje využívané při realizaci SW projektu

2.5.1. phpMyAdmin

Tento volně dostupný nástroj pro správu databáze mySQL je k dispozici na adrese http://www.phpmyadmin.net a jedná se o soubor PHP skriptů, které jsou provozovány na serveru společně s dalšími webovými aplikacemi. Při vývoji na webhostingovém serveru *Webzdarma.cz* (viz. kapitola 3.3.5.) optimálně vyhověl našim požadavkům.

Mimo jiné umožňuje:

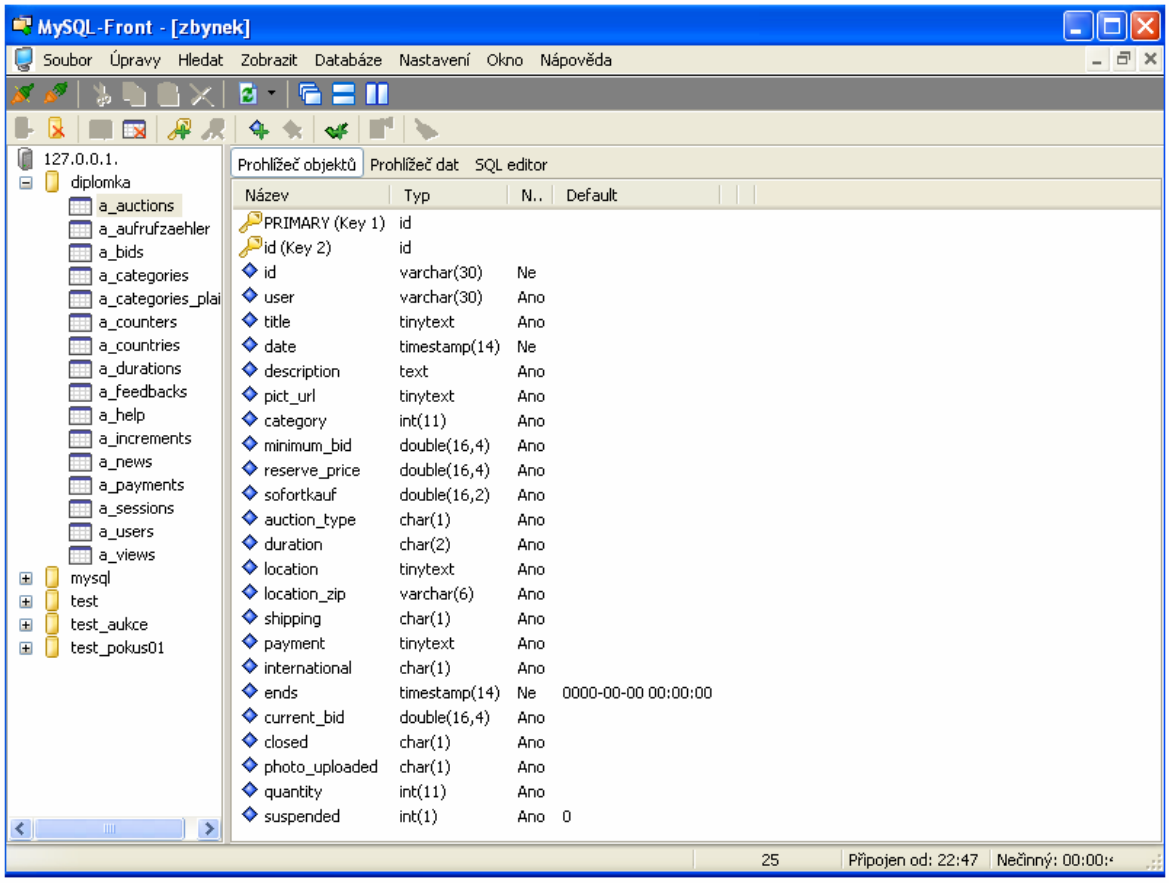
- vytvářet a mazat databázi
- vytvářet, mazat, kopírovat a editovat relace
- vytvářet, mazat a editovat atributy
- spustit libovolný i dávkový SQL příkaz
- spravovat relační klíče
- upravovat oprávnění
- exportovat data do mnoha formátů
- komunikovat v 47 jazycích včetně češtiny



obr. 6 Webové rozhraní phpMyAdmin pro správu databáze mySQL

2.5.2. MySQL-Front

MySQL-Front je uživatelsky příjemné vývojové prostředí pro databáze *MySQL*. Tento program je volně dostupný po dobu 30 dní, poté jsou zablokovány některé funkce, ale i přesto ho lze využít jako velice užitečný prostředek. V porovnání s *phpMyAdmin* (viz. kapitola 2.5.1.) nabízí téměř shodné možnosti, ovšem pro vývoj na lokálním počítači (viz. kapitola 3.3.5.) se zdá byt výhodnější.

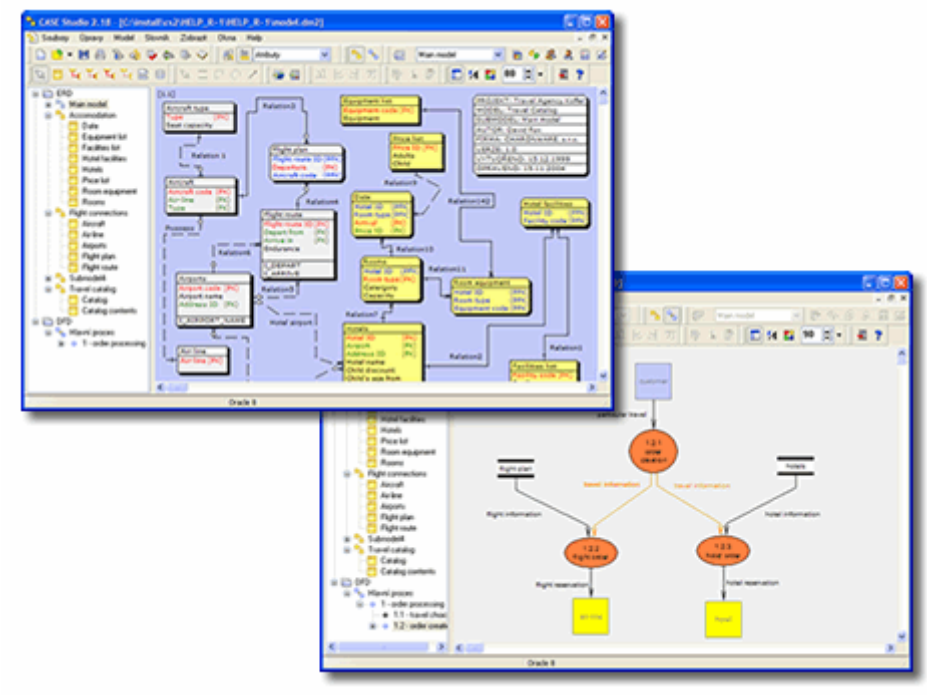


obr. 7 Program MySQL-Front pro správu databáze mySQL

2.5.3. CASE Studio 2

CASE Studio 2 je profesionální software pro vizuální navrhování databázových struktur - jinak řečeno "Database modeler" či "Database designer". Umožňuje vizuálně navrhovat Entitně relační diagramy pro rozličné databáze. Jedná se o ideální software pro společnosti či vývojáře, kteří chtějí pracovat se svými databázemi nejpříjemnějším a časově velice úsporným způsobem. Jedná se o český nástroj firmy CHARONWARE, s.r.o. a je v LITE formě dostupný na WWW adrese http://www.casestudio.com. K jeho výhodám patří:

- Zvýšení efektivnosti práce
- Minimum chyb při vývoji (anebo jejich velmi rychlé odstranění)
- Možnost práce s již existující strukturou databáze (Reverse Engineering)
- Možnost otestovat validitu modelu
- Generování podrobné dokumentace modelu ve formátech HTML a RTF
- Data Flow Diagramy (DFD) - přehledná organizace jednotlivých procesů



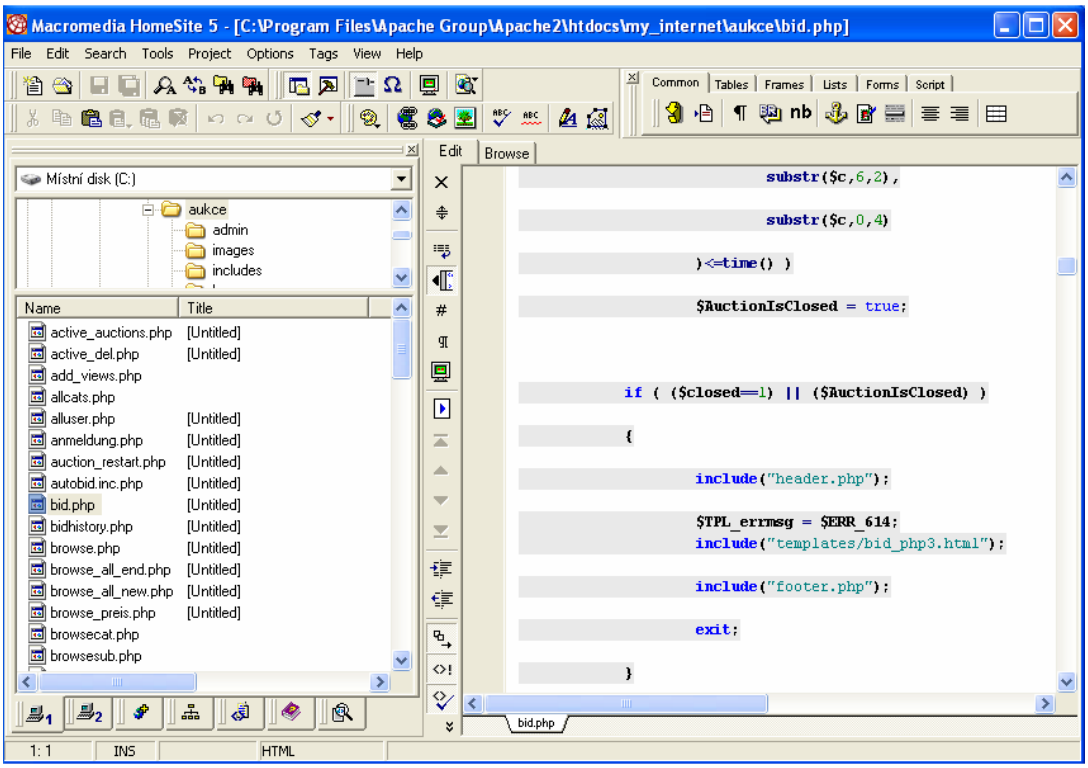
obr. 8 Case Studio 2

2.5.4. HomeSite 5

Program HomeSite od firmy Allaire (http://www.allaire.com) je editor HTML stánek s barevným zvýrazňováním tagů, mnoha užitečnými a upravitelnými klávesovými zkratkami a dalšími funkcemi zrychlujícími návrh stránek. Jeho možnosti jsou velmi bohaté a uspokojí i skutečné profesionály, přesto je jeho prostředí intuitivní a snadno použitelné, takže je vhodný i pro začátečníky.

HomeSite neslouží jen k psaní zdrojového kódu stránky, lze v něm návrh stránky prohlížet přímo v zabudovaném prohlížeči. Od verze 4 je k dispozici i režim WYSIWYG, kde se návrh stránky provádí ve vizuálním režimu a není tak potřeba dokonalá znalost tagů HTML.

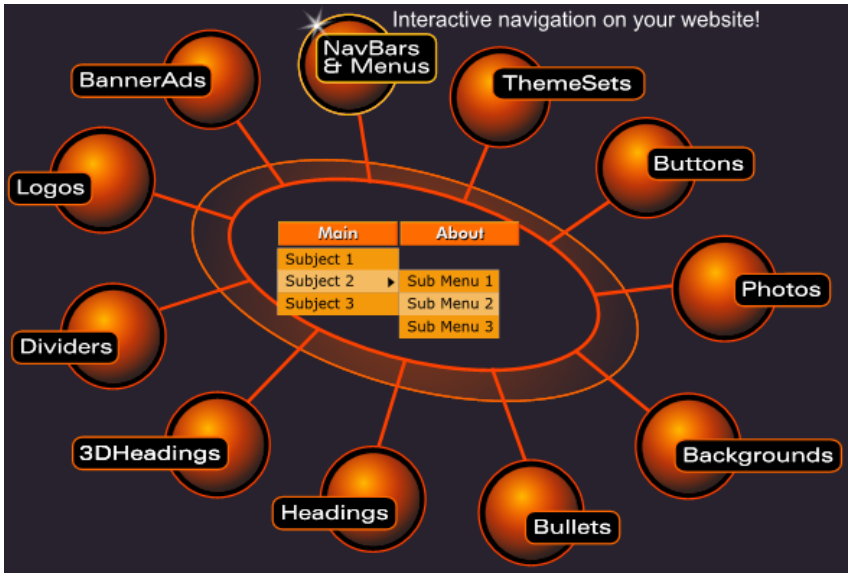
HomeSite je také vhodný na správu celých WWW serverů, umí přistupovat k souborům uloženým na FTP serverech, vytvářet a spravovat projekty, najít nebo nahradit určitý text v libovolném množství souborů najednou a obsahuje řadu dalších pomůcek.



obr. 9 HomeSite 5

2.5.5. Xara Webstyle

Tento produkt nabízí řadu předdefinovaných profesionálních vektorových obrázků (loga, animované bannery, 2D i 3D tlačítka, navigační lišty, pozadí, ...), které lze snadno dle potřeby modifikovat a vytvořit tak vlastní originální grafiku pro WWW. Zkušební 15 denní verze je k dispozici na adrese http://www.xara.com/products/webstyle/



Obr.10 Xara Webstyle 3.0 pro grafický návrh WWW

3. Vlastní realizace projektu - návrh a vývoj databázové aplikace

3.1. Příprava projektu, definice rozsahu

Při vytváření softwarového projektu, je velmi důležité nejprve přesně definovat problémy a požadavky na daný systém. Podceněním této fáze vývoje může později dojít k závažným chybám, které se velmi těžko odstraňují. Udává se, že, přibližně polovina všech chyb, které vzniknou při vývoji aplikace, je způsobena právě nedostatečnou specifikací problému.

Přáním zadavatele, tedy firmy ITI Computers, bylo vytvořit internetovou aplikaci, která by sloužila k obchodování mezi firmami registrovanými v liberecké Agrární komoře. Systém obchodování, jak již bylo zmíněno v úvodu, by měl být formou aukce, tzn. že nabízený produkt bude prodán tomu, kdo nabídne nejvyšší cenu. Jednotlivé firmy (právnické či fyzické osoby) budou zastoupeny nějakým uživatelem, který bude vystupovat pod zvoleným uživatelským jménem.

Důležitým požadavkem bylo, aby systém správně fungoval co nejvíce automaticky bez vnějších zásahů. Součástí aplikace je i tzv. administrátorský přístup, díky němuž může jedna, případně i více osob, aktivně zasahovat do chodu systému. Lze tak např. předčasně ukončovat některé aukce, zamezovat přístup uživatelům, měnit kategorie nabízených produktů apod. Snahou ovšem je, aby zvolený administrátor pouze při spuštění systému nakonfiguroval některá nastavení (kategorie, způsoby plateb,…) a pak již musel zasahovat co nejméně.

Dalším přáním zadavatele bylo, aby aplikace obsahovala nějaký systém hodnocení uživatelů. Po společné konzultaci jsme se rozhodli pro tzv. „hvězdičkové hodnocení„, což znamená, že po každé ukončené aukci, má kupující i prodávající možnost ohodnotit svůj protějšek na stupnici od jedné do pěti hvězdiček. Navíc může ke každému ohodnocení připojit i komentář, ve kterém lze uvést důvody svého rozhodnutí.

V neposlední řadě zadavatel vyžadoval, aby aplikace byla snadno a intuitivně ovladatelná pro uživatele, kteří nemají zkušenosti s podobnými systémy a obecně s informačními technologiemi. Proto jsme se při návrhu ovládání a vzhledu celé aplikace snažili o co největší jednoduchost.

Další konkrétní požadavky již zadavatelem nebyly stanoveny, proto jsme se u některých vzhledových i funkčních prvků nechali inspirovat podobnými

internetovými aukčními systémy, jako jsou např. *aukce.cz, Allegro.pl, a phpAuction.org*.

3.2. Analýza

Vývoj libovolného složitějšího systému vyžaduje použití modelu. Model je schematické popsání reality, které má usnadňovat komunikaci mezi zákazníkem a tvůrcem a mezi členy programátorského týmu.

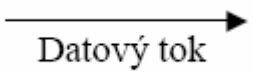
Pro vývoj větších informačních systémů (IS) se vytvářejí dva základní modely: datový a funkční. Datový model je úzce spojen s cílovým databázovým systémem. Dříve se používali modely hierarchické a síťové, nyní to jsou modely relační a objektové. Pro klasické obchodní IS se jako nejvýhodnější stále jeví relační datový model.

Výsledkem analýzy je fyzický datový model a případně také SQL skript, pomocí něhož můžeme vytvořit fyzickou implementaci datového modelu v prostředí DB serveru.

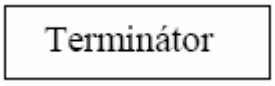
3.2.1. Funkční analýza (DFD)

Cílem funkční analýzy je namodelovat systém jako množinu vzájemně působících funkcí. Nejčastějším vyjádřením jsou DeMarcovy diagramy datových toků DFD (Data Flow Diagram). Systém je pak vyjádřen pomocí procesů spojených datovými toky.

DFD je grafický prostředek zobrazující funkční model systému. DFD vyjadřuje tok dat a jejich transformace, obsahuje popis funkcí, které jsou pro danou transformaci třeba. Nejedná se však o časovou závislost, ale jen o grafické znázornění závislostí mezi jednotlivými kroky, které probíhají v systému. DFD je složen ze čtyř základních prvků:[1]

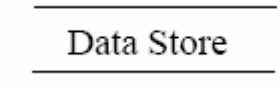


označuje směr přenosu dat – je orientovaný, pojmenovaný a strukturovaný



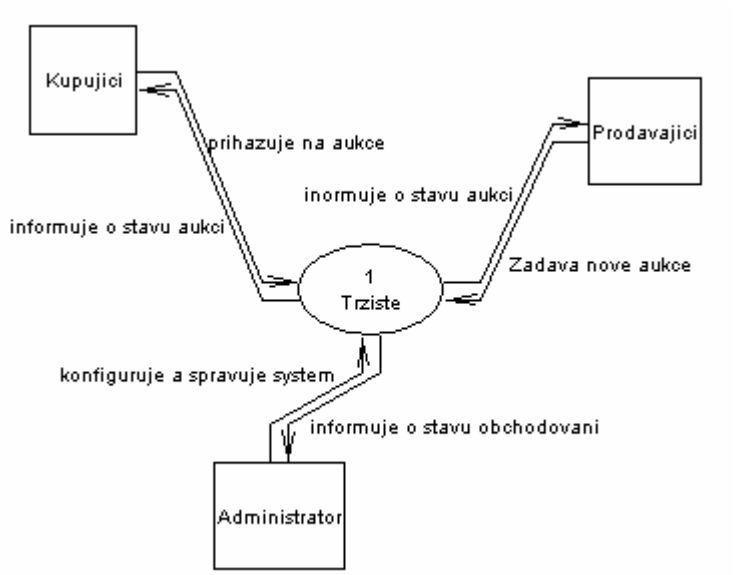
prvky z okolí systému, jsou zdrojem nebo cílem datových toků, nepatří do systému

funkce, proces je transformace dat, při které vstupní data jsou transformována na nějaká data výstupní



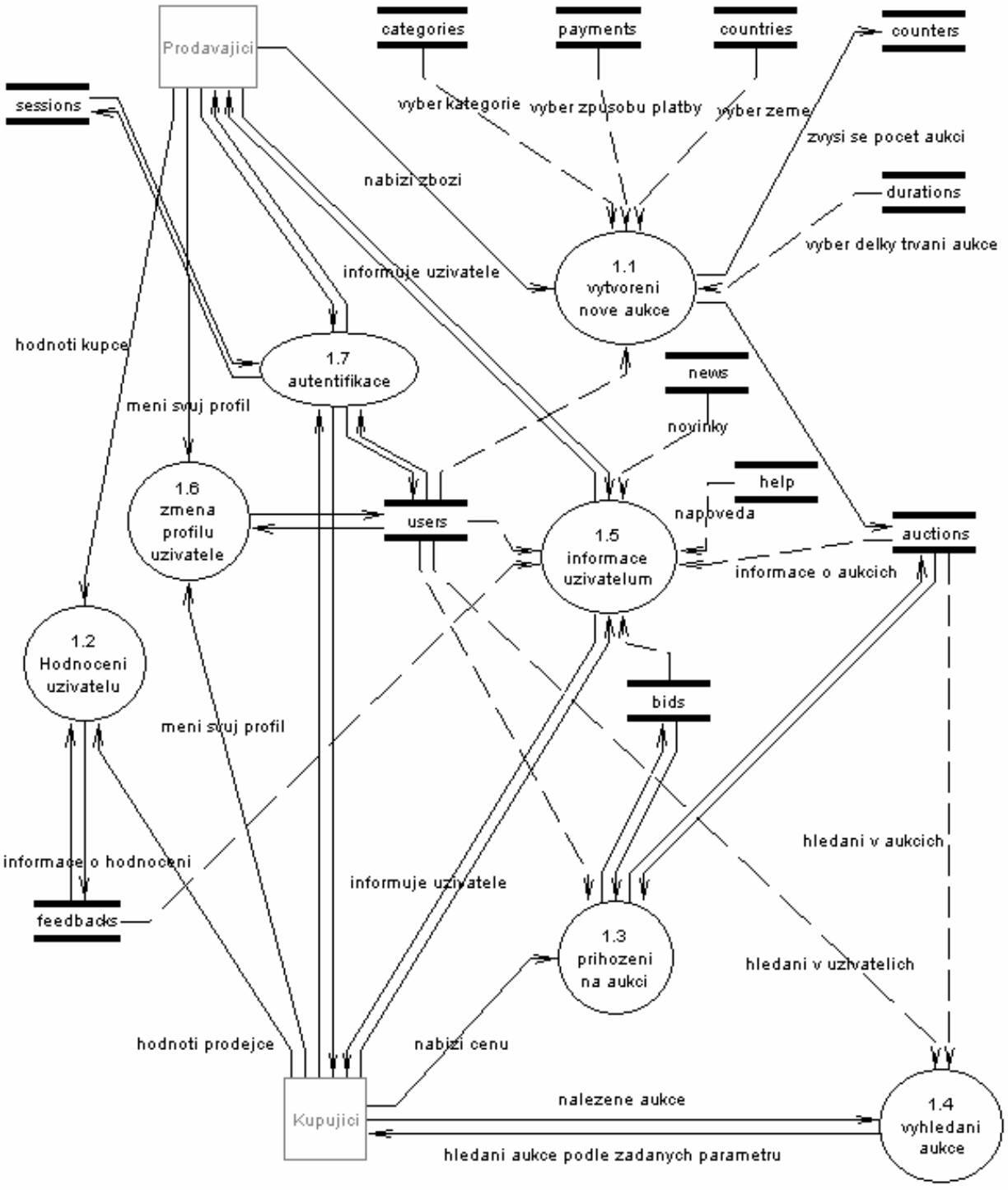
je označení pro libovolný systém uchování dat, pasivní prvek, data v něm nevznikají

DFD je tvořen několika úrovněmi. První úroveň – tzv. kontextový diagram (obr. 11) vyjadřuje, kde bude hranice celého systému a jak bude komunikovat s okolním světem. Další úrovně postupně detailněji popisují funkčnost jednotlivých procesů na výších úrovních. Jedná se tedy o tzv. top - down rozklad, tedy rozklad shora dolů.[2]

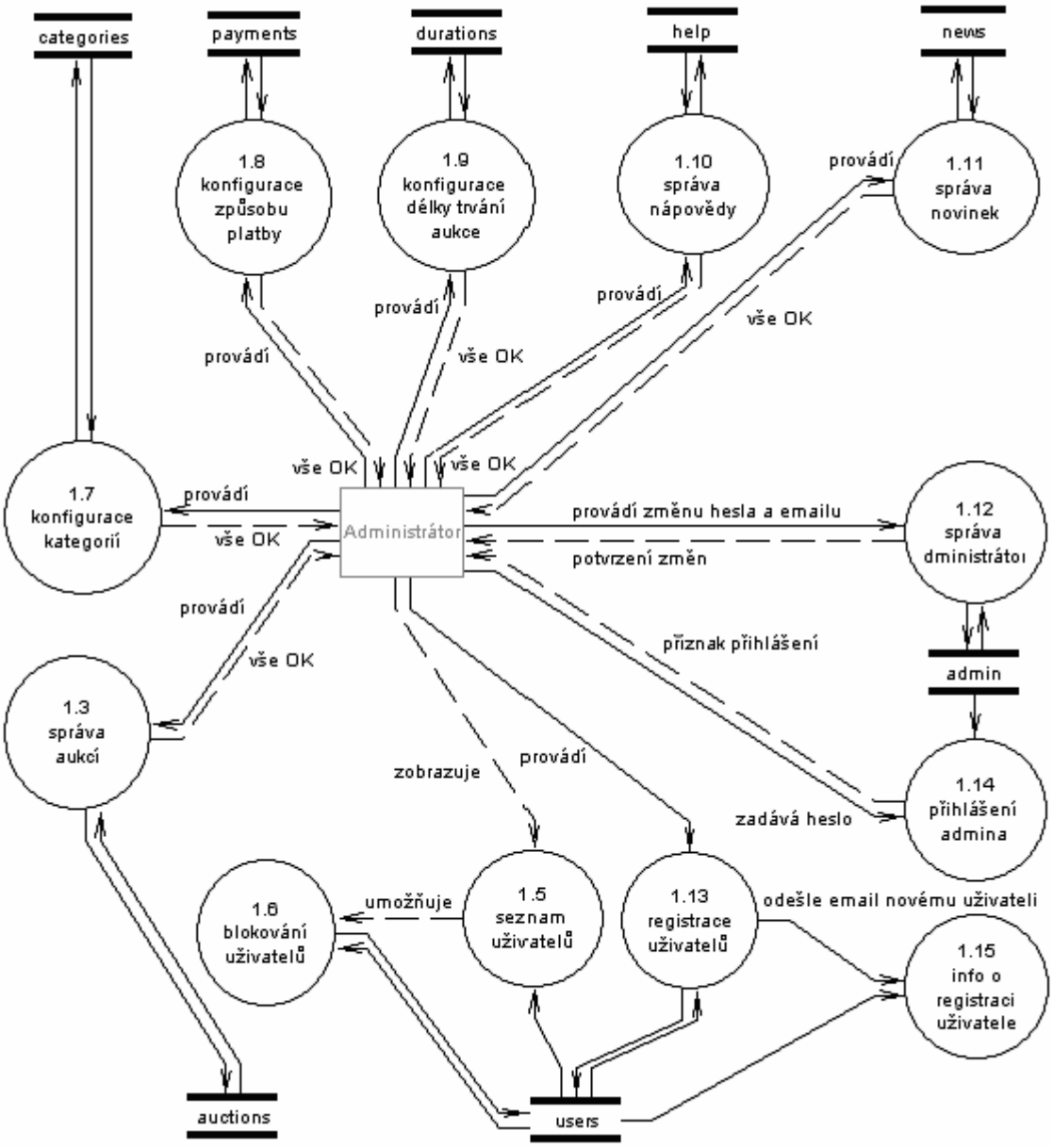


obr. 11 DFD-1.úroveň, kontextový diagram

Na obrázcích 12 a 13 je nakreslena 2.úroveň DFD. Správně by tato úroveň měla být vyjádřena pouze jedním schématem, ovšem pro větší přehlednost jsme zvlášť zobrazili DFD popisující přístup běžného uživatele (kupující nebo prodávající) (obr. 12) a přístup administrátora (obr. 13).



obr. 12 DFD-2.úroveň, uživatelský přístup



obr. 13 DFD-2.úroveň, administrátorský přístup

3.2.1.1. Výsledky funkční analýzy

Z obrázku 11, který znázorňuje napojení systému na vnější prvky, je patrné, že jedním z terminátorů je *kupující,* tzn. běžný uživatel, který má zájem o nabízené zboží, dalším pak *prodávající,* tzn. rovněž běžný uživatel nabízející určitý produkt a v poslední řadě *administrátor*, který spravuje celý systém.

Druhá úroveň (obr. 12) popisuje vnitřní funkci systému s podrobnějším pohledem na jednotlivé funkční bloky, ovšem jen z pohledu běžného uživatele, viz. předchozí kapitola.

Společnými funkčními bloky pro kupujícího i prodávajícího jsou: *autentifikace* (proces 1.7), *změna profilu uživatele* (proces 1.6), *informace uživatelům* (proces 1.5) a *hodnocení uživatelů* (proces 1.2).

Autentifikace znamená, že při jakémkoliv zásahu do systému, který vyžaduje zápis do databáze nebo čtení specifických údajů, je potřeba ověřit totožnost uživatele. Na tento proces je napojen na datastor *sessions* a *users*.

Jednotliví uživatelé si mohou kdykoliv změnit svůj profil, tzn. změnit údaje zadané při registraci, např. heslo, kontaktní údaje apod. K tomu slouží proces *změna profilu uživatele*, napojený na datastor *users*.

Dalším společným procesem je *hodnocení uživatelů*, tím se rozumí možnost kupujícího i prodávajícího navzájem ohodnotit svůj protějšek po ukončení aukce (viz. kapitola 3.1.). Tento proces je napojený na datastor *feedbacks.*

Posledním společným procesem je *informace uživatelům*, ten lze obecně chápat jako veškerý zdroj informací pro kupujícího i prodávajícího. Uživatelé si mohou prohlížet stavy svých aukcí včetně historie příhozů (datastor *bids* a *auctions*), hodnocení ostatních uživatelů (datastor *feedbacks*), dále si mohou přečíst „novinky“ (datastor *news*) nebo nápovědu (datastor *help*), které do systému vkládá administrátor. Velmi důležitou součástí tohoto procesu je automatické rozesílání informačních emailů, o kterém podrobněji pojednává kapitola 3.4.8.

Hlavní funkcí prodávajícího je nabídnout nějaký produkt, tzn. vytvořit novou aukci (proces 1.1). Tento proces je samozřejmě napojen na datastor *auctions* a *users*. Při vytváření nové aukce (kapitola 3.4.4) je třeba vyplnit některé povinné údaje, jako jsou kategorie, způsob platby, země a doba trvání aukce. Hodnoty, které je možno zadat pro tyto položky, jsou přednastavené administrátorem a jsou uloženy v datastorech *categories, payments, countries* a *durations*. Po vytvoření aukce se pochopitelně zvýší celkový počet aukcí, toto „počítadlo“ je uloženo v datastoru *counters.*

Mezi funkce (procesy) přidělené kupujícímu patří zejména přihození na nějakou aukci (proces 1.3). Tento proces musí být logicky napojen na datastory *auctions, users* a *bids*, ve kterém jsou uloženy veškeré příhozy na všechny aukce.

Jestliže uživatel (kupující) hledá určitý produkt, tzn. aukci, tak k tomu slouží funkce vyhledání aukce (proces 1.4) (viz. kapitola 3.4.6.). Protože lze vyhledávat jednak v samotných aukcích, tak i ve jménech uživatelů, je tento proces svázán s datastory *auctions* a *users*.

Podrobnější popis 2. úrovně z hlediska administrátora není součástí této práce, protože tuto část vytváří kolega Tobolka (viz. Úvod)

3.2.2. Datová analýza (ERD)

Datová analýza má za úkol popsat vztahy mezi daty, která v systému existují, a jejich strukturou, tzn. graficky znázornit daný relační model. Právě k tomuto účelu slouží entitně relační diagramy (ERD).

Všechny typy ERD se skládají, jak samotný název ukazuje, z entit a relací. Entita je v ERD diagramu reprezentována obdélníkem, ve kterém je uvedeno jméno entity. Reprezentuje skupinu nebo množinu objektů (věcí) v reálném světě, o kterých chceme uchovávat data. Entita je složená z prvků (atributů), které ji charakterizují. Entity jsou mezi sebou navzájem propojeny relacemi. Relace (vztah) reprezentuje množinu vazeb mezi entitami. U každého vztahu se definuje tzv. *kardinalita*, která vyjadřuje mocnost vztahu entit. Rozlišujme tři základní mocnosti: [2]

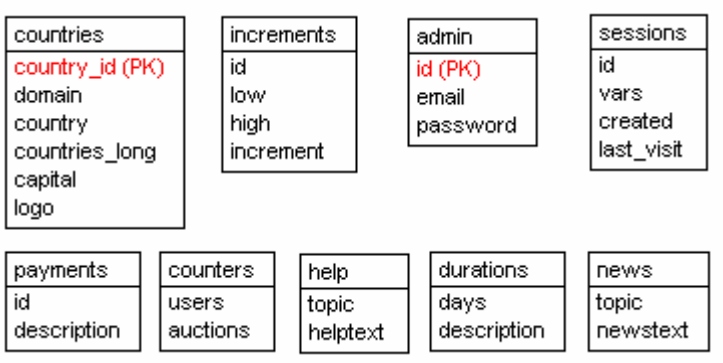
- 1:1, vyjadřuje skutečnost, že obě entity jsou ve vztahu zastoupeny pouze jednou
- 1:N, vyjadřuje skutečnost, že k jednomu výskytu entity A existuje více výskytů entity B. Jedná se o nejčastější případ vztahu dvou entit
- M:N , vyjadřuje skutečnost, že k jednomu výskytu entity A existuje více výskytů entity B a naopak. Tento vztah není realizovatelný ve fyzickém návrhu databáze a je třeba jej nahradit vazební entitou, která vztah převede na dva vztahy 1:N

Dále se definuje tzv. *volitelnost* vztahu, která vyjadřuje zda je účast entit ve vztahu nutná nebo zda je volitelná a nemusí existovat. [2]

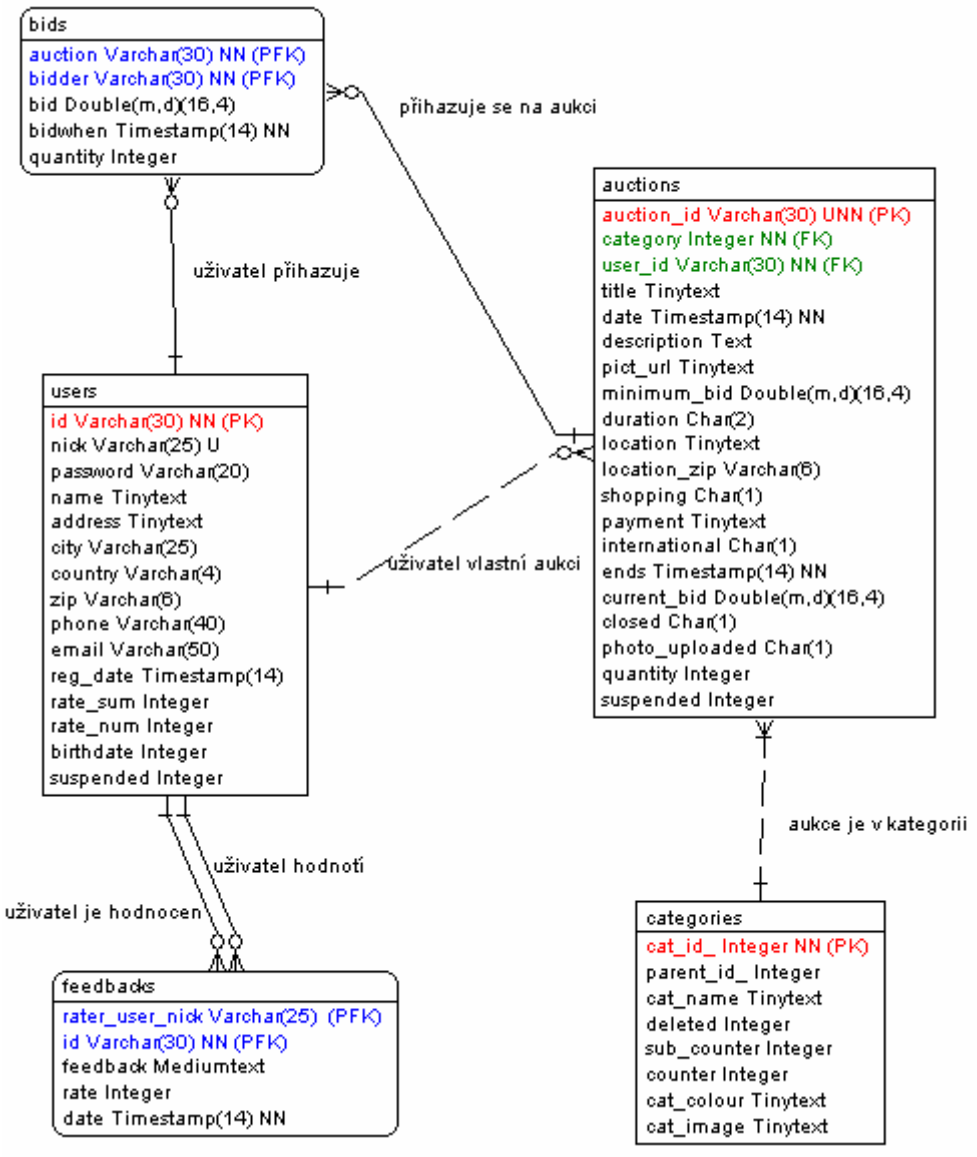
Datový model vyjadřuje vztah mezi typy entit bez popisu funkcí, které tyto data vytvářejí nebo je používají.[2]

ERD model našeho systému je na obrázku 14 a 15. Na obrázku 14 jsou zobrazeny entity mezi kterými neexistují žádné vztahy.

Při vytváření modelu, tzn. jednotlivých entit, byla nutná spolupráce s kolegou Tobolkou, protože výsledná datová struktura je společná pro uživatelskou i administrátorskou část systému.



obr. 14 ERD – 1.část



obr. 15 ERD – 2.část

3.2.2.1. Výsledky datové analýzy

Výsledkem datové analýzy je logický datový model reprezentovaný ERD diagramem. V našem případě každá entita odpovídá jedné fyzické tabulce v databázi.

Hlavní entitou je *auctions*, ve které jsou uloženy údaje o jednotlivých aukcích. Entita *users* obsahuje záznamy o registrovaných uživatelích. Vztah k entitě *auctions* je 1:N, protože jeden uživatel může vlastnit více aukcí.

Entita *categories* slouží k uložení kategorií. Každá aukce musí být zařazena v nějaké kategorii, vztah k entitě *auctions* je 1:N, protože více aukcí může být v jedné kategorii.

Historie příhozů na veškeré aukce je zastoupena entitou *bids*. Vztah k entitám *users* a *auctions* má v obou případech N:1. Tyto relace jednoznačně definují jednotlivé záznamy v této entitě, protože každý „příhoz“ je určen jednak aukcí, na kterou byl přihozen, a uživatelem, který přihodil.

Poslední významnější entitou je *feedbacks*, která obsahuje vzájemné hodnocení uživatelů. Tato entita má dvě vazby na entitu *users*, obě ve vztahu N:1. Opět tyto vztahy jednoznačně definují jednotlivé záznamy v entitě *feedbacks*, protože každé hodnocení je dáno uživatelem, který hodnotí, a uživatelem, který je hodnocen.

Co se týče ostatních entit (obr. 14), tak entity *payments, countries* a *durations* slouží k uložení přípustných hodnot pro způsob platby, zemi a dobu trvání aukce (viz. kapitola 3.2.2.1.). Nabízí se otázka, proč nejsou tyto entity napojeny na tabulku *auctions*. Důvod je ten, že údaje obsažené v těchto entitách nejsou příliš významné a je proto lepší uložit hodnoty v tabulce *auctions* přímo jako text, resp. číslo, a ne jako odkazy do jiných tabulek. Tím se sníží nároky na databázový server, např. při zobrazení detailního pohledu na aukci se údaje čtou pouze z tabulky *auctions* a ne již ze zmiňovaných tabulek.

Entita *counters* slouží jako „počítadlo“ uživatelů a aukcí. V entitách *help* a *news* je uložena nápověda, resp. novinky, obě má právo měnit pouze administrátor. V entitě *increments* jsou uloženy minimální příhozy v závislosti na aktuální ceně nabízeného produktu. Je-li např. aktuální nabídka 10000 Kč není možné přihodit 1 Kč, ale minimálně 100 Kč. Právo určovat tyto hodnoty má opět

pouze administrátor. Pro uložení hesel administrátor(a)ů je k dispozici entita *admin*.

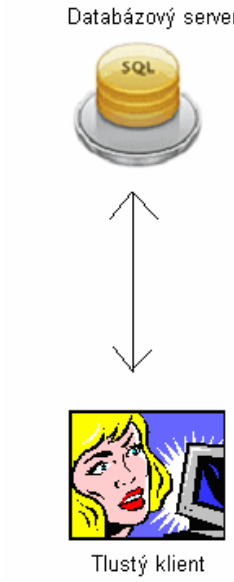
Entita *sessions* je určená k ověření autentičnosti uživatelů ve speciálních případech a bude podrobněji vysvětlena v kapitole 3.7.

Výsledný skript v jazyce SQL, který vytváří veškeré databázové objekty včetně několika předdefinovaných záznamů je součástí přílohy.

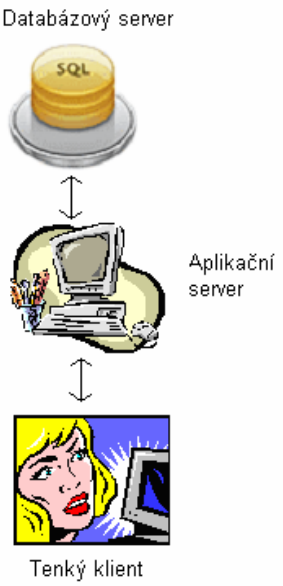
3.3. Návrh

3.3.1. Návrh architektury systému

Při návrhu architektury systému existují dvě možnosti: dvouvrstvá architektura (obr. 16) a třívrstvá architektura (obr. 17)



Obr. 16 Dvouvrstvá architektura



Obr. 17 Třívrstvá architektura

Dvouvrstvá architektura odděluje databázový server a klientskou aplikaci, kterou je tlustý klient, třívrstvá je dělena na vrstvu databázového serveru, aplikačního serveru doplněného vhodným skriptovacím jazykem a tenkého klienta, který má korektně zobrazovat data odeslaná aplikačním serverem. U třívrstvé architektury jsou všechny skripty aplikace uloženy na aplikačním serveru, kde je lze snadno aktualizovat, aniž by bylo nutné zasahovat do nastavení u uživatele. Naproti tomu dvouvrstvá architektura má všechen programový kód implementován v podobě klientské aplikace tzv. „tlustého klienta“ přímo na PC uživatele [2].

Protože se předpokládá, že systém bude používat v nejbližší budoucnosti okolo 200 firem (uživatelů), později možná více, je téměř nemožné, aby

byla aplikace instalována u každého uživatele individuálně. Proto jsme zvolili třívrstvou architekturu.

3.3.2. Návrh aplikačního serveru

Vzhledem k požadavku zadavatele na nulové náklady připadala v úvahu v podstatě jediná možnost – aplikační server *Apache* (viz. kapitola 2.1.). Jedná se o robustní a stabilní server s nativní podporu skriptovacího jazyka PHP, který plně uspokojí požadavky systému. Nicméně nejvýznamnějším faktorem byla jeho nulová cena.

3.3.3. Návrh databázového serveru

Stejně jako u výběru aplikačního serveru i zde hrála největší roli cena. Z tohoto důvodu jsme zvolili volně dostupný server *MySQL*. Je to poměrně výkonný a velmi rychlý stroj. Jeho nevýhodou je absence některých funkcí, např. transakcí, které ale nejsou pro naši aplikaci nutné.

Dalšími možnostmi, ovšem již komerčními, byly Microsoft SQL Server a databázový stroj Oracle. Jejich cena se ale pohybuje v řádu statisíců, což je pro zadavatele nepřijatelné.

3.3.4. Návrh skriptovacího jazyka

Po zvolení aplikačního serveru *Apache*, byla volba skriptovacího víceméně jednoznačná a to PHP. Je to volně šiřitelný produkt (freeware), který si nejlépe rozumí právě se serverem *Apache*, ale funguje i pod *Microsoft IIS.* Výhodou PHP je nezávislost na platformě, tzn. že ho lze použít ve Windows i v Unixu. V neposlední řadě hovoří v jeho prospěch podobná syntaxe s jazykem C/C++, který je značně rozšířen mezi programátory.

3.3.5. Návrh vývojového prostředí

Jako vývojové prostředí jsme si vybrali kromě lokálního počítače i webhostingový server *Webzdarma.cz.*

Na lokálním počítači bylo nejprve nutné nainstalovat server *Apache*, dále skriptovací jazyk PHP a nakonec databázový server *MySQL.*

Důvod proč jsme použili i server *Webzdarma.cz ,* je ten, že chování aplikace na cílovém serveru zadavatele se přece jenom více blíží chování na tomto internetovém serveru než na lokálním počítači. Lze tak lépe provádět testování.

Na českém trhu i v zahraničí existuje značné množství webhostingových serverů, které nabízejí podobné služby jako Webzdarma.cz, ovšem většina z nich si účtuje minimální poplatky narozdíl od tohoto serveru. Poskytovatel nabízí všechny námi požadované technologie zcela zdarma. Mezi základní vlastnosti serveru Webzdarma.cz patří:

- uživatelský prostor o velikosti 50MB pro WWW stránky
- e-mail schránka o velikosti 5MB s přístupem přes webové rozhraní
- adresu ve tvaru domény 3. řádu s možností výběru z 16 domén 2. řádu
- databázi MySQL o velikosti 5MB
- aktualizaci stránek pomocí FTP nebo webové rozhraní
- podporu PHP skriptu
- uživatelskou podporu prostřednictvím diskusního fóra nebo e-mailu
- phpMyAdmin pro správu mySQL databáze

3.3.6. Návrh webového prohlížeče pro koncového uživatele

Celý vývoj a optimalizace systému probíhala v prohlížeči Internet Explorer od firmy Microsoft, protože tento produkt je bezesporu nejvíce rozšířen ve světě a tedy i v ČR.

Testování ukázalo, že aplikace pracuje téměř bez problémů i v prohlížeči Firefox od společnosti Mozilla. Až na drobné grafické odchylky, které lze přehlédnout, se aplikace chová stejně jako v Exploreru. V poslední době se tento program stává čí dál tím více oblíbenější, zejména kvůli bezpečnosti, která je vytýkána Exploreru.

3.4. Vývoj uživatelské aplikace

Uživatelská aplikace je určena pro běžné uživatele, tzn. pro kupující a prodávající. Při návrhu grafického provedení a ovládání aplikace jsme se snažili především o jednoduchost, tak aby byl systém přizpůsoben potřebám uživatelů, kteří nemusejí být seznámeni s prací v prostředí internetu. Veškeré formuláře byly zjednodušeny na nezbytně nutné položky a správnost vložených údajů je zkontrolována ještě před odesláním.

Součástí aplikace je nápověda, která je rozdělena do tématických okruhů. Tyto témata spravuje (přidává,maže,edituje) administrátor. Implicitně je při instalaci systému k dispozici stručná obecná nápověda „Jak používat Tržiště“.

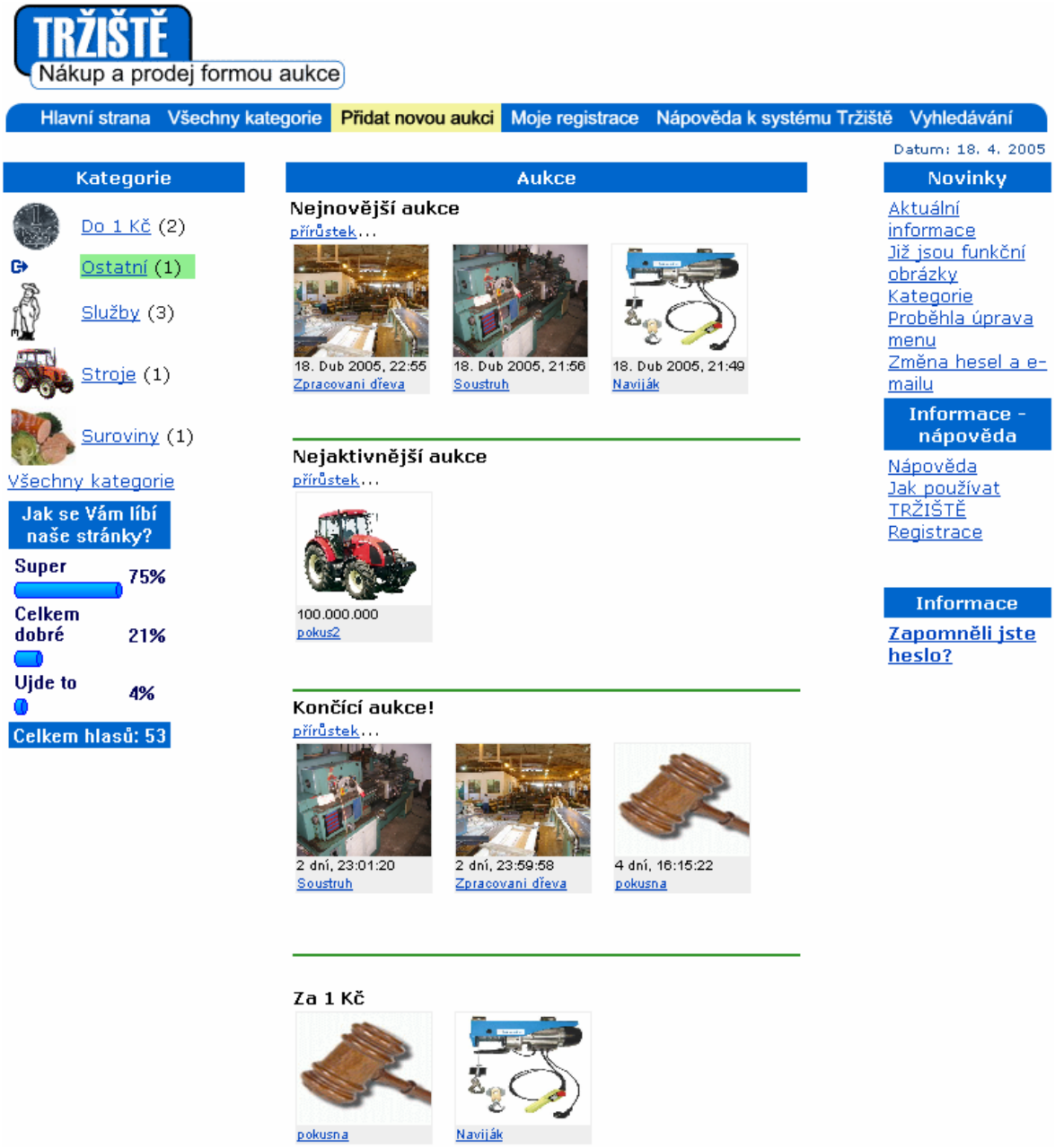
3.4.1. Hlavní strana

Při spuštění aplikace se jako první objeví „hlavní strana“ (obr. 18). Je vidět, že v horní části obrazovky, pod logem systému Tržiště, se nachází hlavní navigační menu aplikace. Toto menu se objevuje na stejném místě ve všech „podstránkách“ aplikace. Jsou na něm umístěny odkazy na důležité funkce systému. Bylo navrženo v programu Xara Webstyle (viz. kapitola 2.5.5.), který vygeneroval všechny soubory definující funkci celého menu v jazyku JavaScript.

Vlevo pod hlavním menu se nacházejí jednotlivé kategorie včetně počtu běžících aukcí .

Uprostřed obrazovky jsou zobrazeny některé vybrané aukce podle specifických kriterií. Jedná se o nejnovější aukce, nejaktivnější aukce, tzn. aukce, na které se nejvíce přihazuje, dále končící aukce a aukce u kterých byla stanovena vyvolávací cena na 1 Kč.

V pravém panelu se jsou pak umístěny informativní odkazy, mezi které patří novinky, nápověda a odkaz na řešení problému při zapomenuti uživatelského hesla.



obr. 18 Hlavní strana

3.4.2. Kategorie

Každá aukce musí být zařazena v nějaké kategorii, které definuje administrátor. Je to z důvodu větší přehlednosti a snazšímu vyhledávání nabízených produktů. Lze vytvářet i podkategorie, např. kategorie obiloviny je podkategorií kategorie suroviny (obr. 19). V případě, že by uživateli nevyhovovala

žádná z předdefinovaných kategorií, může vždy zařadit svůj produkt do kategorie „Ostatní“, která je právě k tomuto účelu vytvořena.

obr. 19 Kategorie

3.4.3. Přihození na aukci

Po „kliknutí“ na jakoukoliv aukci se zobrazí její detailní popis (obr. 20) a odtud lze na danou aukci přihodit.

Po zadání cenové nabídky a odeslání formuláře na přihození se nejprve zkontroluje výše nabídky. Nabídka musí být alespoň taková, aby se zvýšila aktuální cena o minimální příhoz (viz. kapitola 3.2.2.1.) nebo, v případě, že se jedná o první přihození, se musí alespoň rovnat vyvolávací ceně. Je-li toto splněno, systém si vyžádá uživatelské jméno a heslo, tak aby mohli přihazovat jen registrovaní uživatelé. Pokud je i toto v pořádku, systém nakonec ověří aktuální stav aukce, tzn. zda již aukce nebyla ukončena či zrušena administrátorem a nestalo-li se tak, je uživatelova žádost (nabídka) definitivně přijata. Uživateli je poté odeslán potvrzující email (viz. kapitola 3.4.8.).

TRŽIŠTĚ
Nákup a prodej formou aukce

Hlavní strana | Všechny kategorie | Přidat novou aukci | Moje registrace | Nápověda k systému Tržiště | Vyhledávání

Datum: 19. 4. 2005

Soustruh

Stroje / Ostatní stroje

Prohlédnout
Popis

Aktuální cena:	**0**
Zbývající čas:	**1 dní, 23:22:02**
Prodávající:	martin Kontakt s prodávajícím Zhodnocení prodávajícího Všechny aukce prodávajícího
Příhozů # :	0
Vyvolávací cena (start):	25.000
Minimální příhoz k aktuální ceně:	100

Nabídka

Aktuální cena:	0
Minimální příhoz k aktuální ceně:	100
VAŠE NABÍDKA:	Kč

(Minimální nabídka:25.000)
Odeslat

Popis

ok

1

Země: Slovensko(11150)
Náklady na doručení: Kupující platí náklady spojené s dopravou, Zboží lze zaslat mimo uzemí ČR
Způsob platby: Bankovní převod, Hotově, Jinak dohodou

Počátek aukce:18. Dub 2005, 21:56
Konec aukce:21. Dub 2005, 21:56
ID aukce:426410724

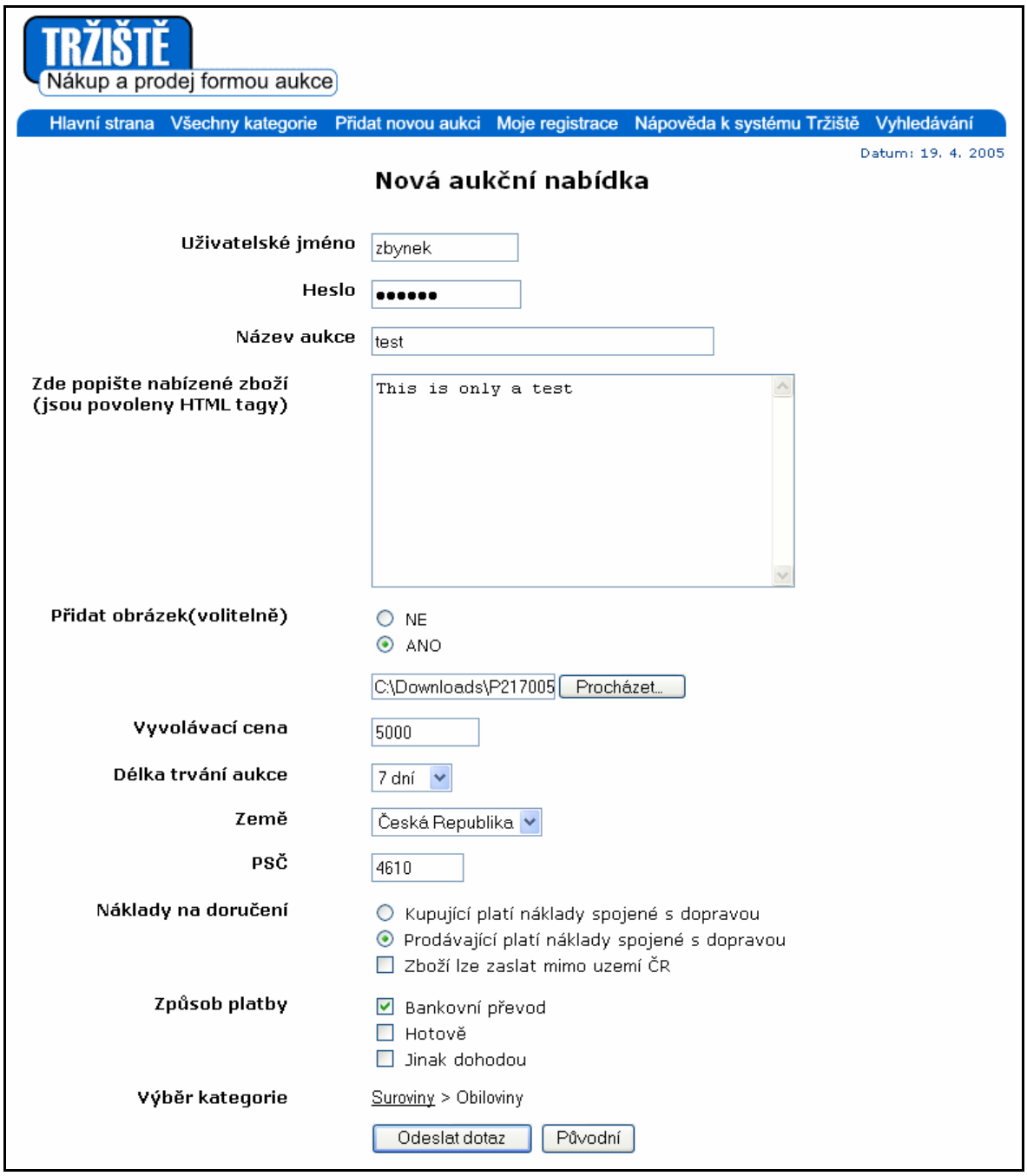
Kategorie: Stroje / Ostatní stroje

obr. 20 Detailní pohled na aukci - přihození

3.4.4. Přidání nové aukce

Má-li uživatel zájem prodat určitý produkt, tzn. vytvořit novou aukci, tak k tomuto účelu slouží položka *Přidat novou aukci* z hlavního menu aplikace.

Po vybrání kategorie, případně podkategorie, se zobrazí formulář pro zadání parametrů k nové aukci (obr. 21).



obr. 21 Vytvoření nové aukce

Protože tato služba je přístupná samozřejmě pouze pro registrované uživatele, je nutné zadat uživatelské jméno a heslo. Dále je třeba, aby uživatel zadal:

- *název aukce*
- *popis* nabízeného produktu
- zda připojí *obrázek*
- *vyvolávací cenu*
- *délku trvání aukce*
- *stát* (Česká republika, Slovensko)
- *PSČ*
- Kdo platí *náklady spojené s dopravou*
- *způsob platby*

Po odeslání dotazu (formuláře) systém ověří, jestli jsou vyplněny všechny položky, u některých zkontroluje i jejich syntaktickou správnost (např. PSČ) a v případě, že je vše v pořádku, zobrazí se náhled nové aukce. Uživatel má tak možnost naposledy překontrolovat svoje zadané údaje a rozhodnout, zda ještě něco upraví nebo definitivně potvrdí svojí nabídku (aukci). Po potvrzení se v databázi fyzicky vytvoří nová aukce a uživateli je odeslán potvrzující email (viz. kapitola 3.4.8.).

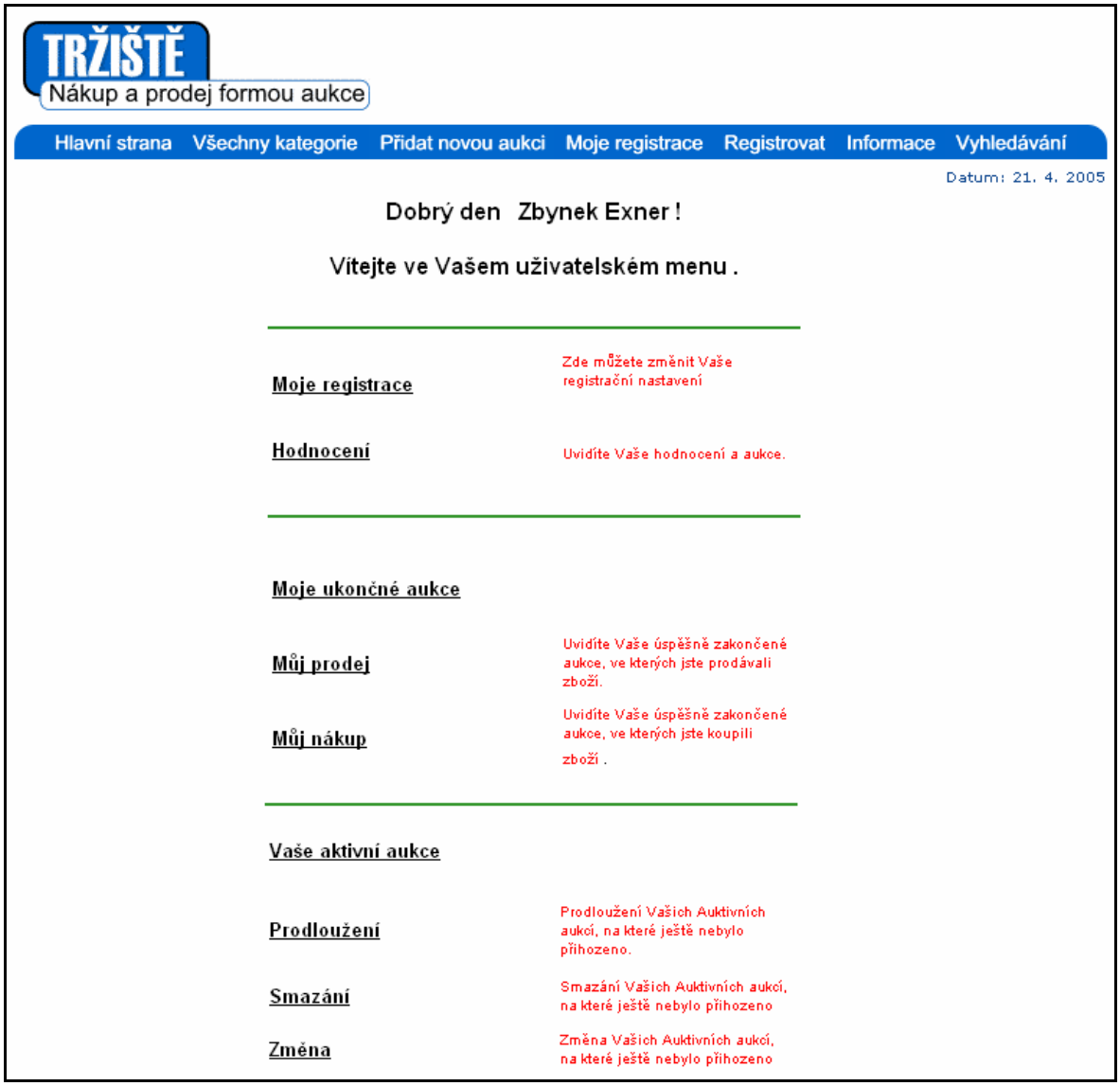
3.4.5. Moje registrace

Aplikace umožňuje uživateli měnit si svoje registrační údaje a prohlížet si všechny svoje běžící i ukončené aukce z hlediska prodávajícího i kupujícího. K tomu je určen odkaz v hlavním menu *Moje registrace.*

Nejprve je pochopitelně vyžádáno uživatelské jméno a heslo a poté se již uživatel dostane do svého „uživatelského menu“ (obr. 22). Zde má k dispozici následující položky:

- *Moje registrace* – umožňuje uživateli měnit svoje registrační údaje, např. přístupové heslo, adresu, email apod.
- *Hodnocení* - zobrazuje hodnocení od ostatních uživatelů
- *Moje ukončené aukce* – vypíše seznam všech ukončených aukcí, kterých se uživatel účastnil, tzn. kde prodával i kupoval
- *Můj prodej* – zobrazí seznam úspěšně (existuje-li kupec) ukončených aukcí, ve kterých uživatel figuroval jako prodávající

- *Můj nákup* – zobrazí ukončené aukce, kde uživatel „vyhrál“, tzn. že má právo k nákupu daného zboží
- *Vaše aktivní aukce* – vypíše seznam všech aktivních (běžících) aukcí, kterých se uživatel účastní (prodává nebo kupuje)
- *Prodloužení* – umožní uživateli prodloužit aukce o zvolený počet dnů, lze tak učinit pouze u aukcí, které skončily a nikdo nepřihodil nebo u aukcí, které běží a doposud nikdo nepřihodil
- *Smazání* – uživatel může smazat (zrušit) svoje běžící aukce, ale pouze v případě, že na ně nebylo již přihozeno
- *Změna* – uživatel může změnit parametry svých aukcí, na které ještě nebylo přihozeno, např. doplnit popis, změnit působ platby apod.

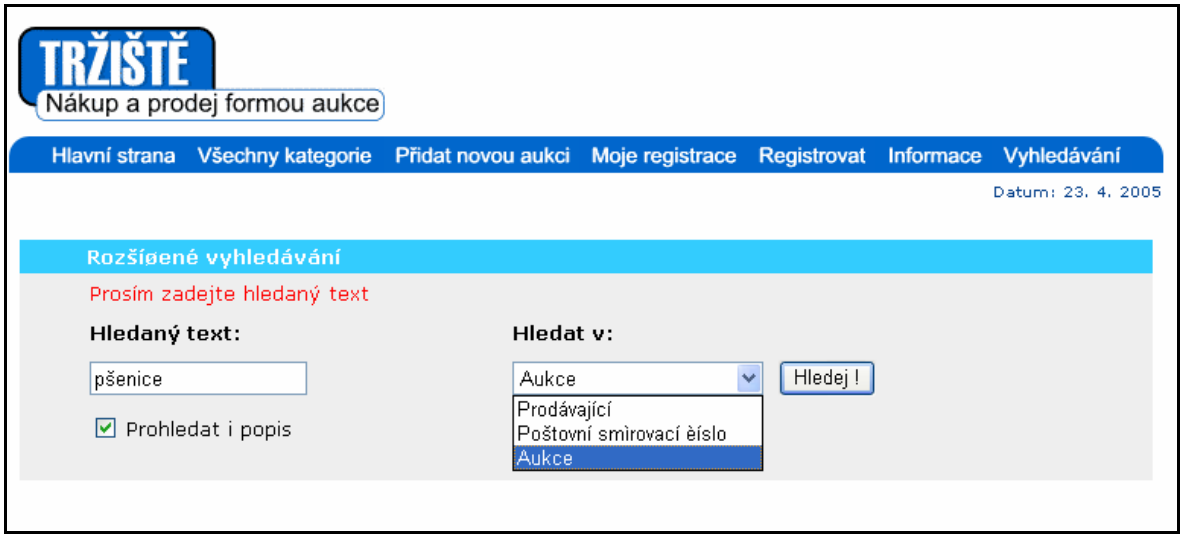


obr. 22 Moje registrace – uživatelské menu

3.4.6. Vyhledávání

Pro vyhledání konkrétních aukcí slouží položka *Vyhledávání* v hlavním navigačním menu.

Hledat lze, vedle názvů aukcí, i ve jménech prodávajících a jejich poštovních směrovacích číslech. Dále je možno prohledávat zadaný text i v popisech jednotlivých aukcí (obr. 23).



obr. 23 Vyhledávání

3.4.7. Novinky, informace a nápověda

V pravé části hlavní strany (obr. 18) jsou k dispozici odkazy na novinky a nápovědu.

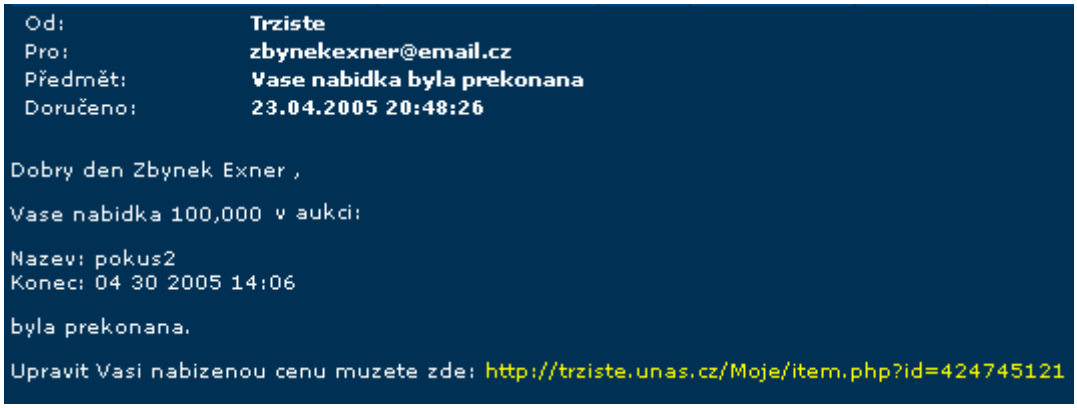
Novinky jsou textová sdělení, které může průběžně přidávat i odstraňovat administrátor. Příkladem může být informace o tom, že byly přidány nové kategorie apod.

Nápovědu rovněž spravuje administrátor, je rozdělena do tématických okruhů, např. obecné informace o tom , jak používat systém tržiště nebo jak se registrují noví uživatelé apod.

3.4.8. Automatické rozesílání emailů

Jednou z předností našeho systému je automatické rozesílání informativních emailů pro kupující a prodávající. Jedná se o to, že při jakékoliv změně stavu aukce se odešlou upozorňující emaily zainteresovaným uživatelům. Typy takovýchto emailů jsou v následujícím přehledu:

- po vytvoření aukce – Potvrzení prodávajícímu, že aukce byla úspěšně spuštěna.
- po ukončení aukce – Informuje prodávajícího i v případě, že nikdo nepřihodil, dále kupujícího, který „vyhrál“ danou aukci.
- po přihození na aukci – Informuje prodávajícího a kupujícího, který přihodil, jako potvrzení jeho příhozu. Dále se odešle email kupujícímu, který měl až doposud nejvyšší nabídku, je to upozornění, že jeho nabídka již byla překonána. Tento email pak obsahuje mimo jiné i přímý URL odkaz na danou aukci, takže lze jednoduše a rychle opět zvýšit svojí peněžní nabídku (obr.24).
- po zaregistrování nového uživatele – Potvrzení o úspěšné registraci.


Od: Trziste
Pro: zbynekexner@email.cz
Předmět: Vase nabidka byla prekonana
Doručeno: 23.04.2005 20:48:26

Dobry den Zbynek Exner ,

Vase nabidka 100,000 v aukci:

Nazev: pokus2
Konec: 04 30 2005 14:06

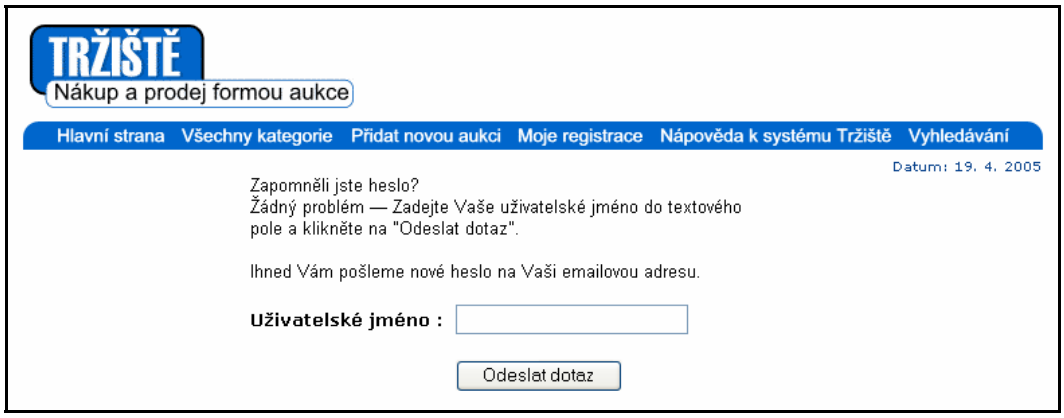
byla prekonana.

Upravit Vasi nabizenou cenu muzete zde: http://trziste.unas.cz/Moje/item.php?id=424745121

Obr.24 Email, upozorňující, že nabídka byla překonána

3.4.9. Problém při zapomenutí hesla

Nastane-li situace, že uživatel zapomene svoje přihlašovací heslo, je možné pomocí odkazu „*Zapomněli jste heslo*“ na hlavní straně vpravo dole (obr.18), vyžádat si heslo nové (obr.25). Systém vygeneruje nové heslo jako náhodný textový řetězec a odešle ho uživateli na jeho emailovou adresu. Uživatel si poté samozřejmě může toto heslo změnit ve menu *Moje registrace* (kapitola 3.4.5.).



obr. 25 Formulář pro vyžádání nového hesla

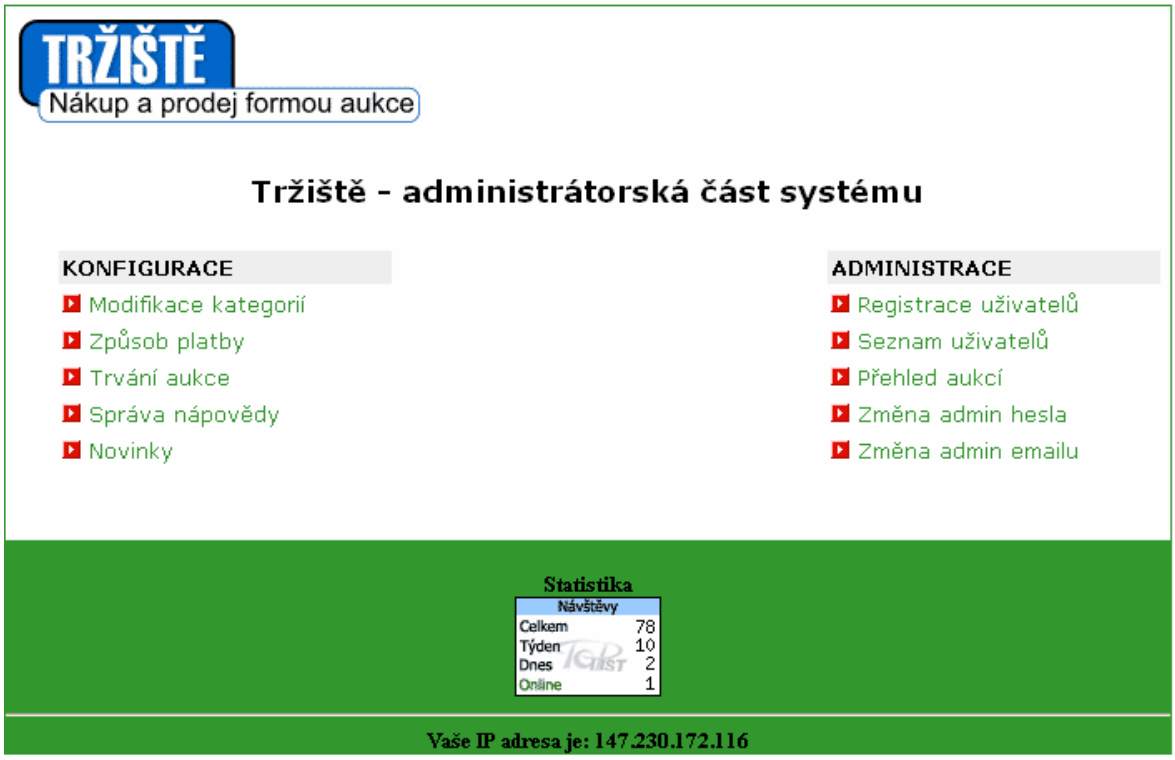
3.4.10. Aktualizace stavů aukcí

Aby systém správně fungoval je zapotřebí kontrolovat pravidelně stavy aukcí, to znamená zjišťovat zda by již nějaká aukce neměla být z časového hlediska ukončena. V takovém případě je potom nutné aktualizovat v databázi tabulky *auctions, categories* (sníží se počet aktivních aukcí v dané kategorii) a *counters* (sníží se celkový počet aukcí). Dále je zapotřebí odeslat email prodávajícímu a existuje-li tak i kupujícímu, který „vyhrál“.

Všechny tyto operace se provádějí pomocí souboru *cron.php.* Tento skript se spustí vždy při zobrazení hlavní strany, tzn. načte-li se stránka *index.html* resp. *index.php.* V praxi je ovšem lepší, aby se tento skript spouštěl daleko častěji. Je tedy nutné nakonfigurovat cílový server, na kterém je uložen skript, tak aby docházelo automaticky ve zvoleném časovém intervalu ke spuštění tohoto skriptu.

3.5. Vývoj administrátorské aplikace

Jak již bylo zmíněno v úvodu, tuto část projektu vytváří kolega Martin Tobolka. Jedná se o součást aplikace, která umožní pověřené osobě provádět konfiguraci a správu celého systému. Tato osoba – administrátor má za úkol definovat kategorie, způsoby plateb, trvání aukcí, dále může upravovat nápovědu a přidávat novinky. K dispozici má samozřejmě veškeré přehledy uživatelů a aukcí, které může podle potřeby upravovat. V současné době má jako jediný možnost registrovat nové uživatele. Hlavní strana administrátorského subsystému je na obrázku 26.



obr. 26 Hlavní strana administrátorského subsystému

3.6. Struktura souborů a adresářů aplikace

Při vytváření aplikace jsme pro větší přehlednost navrhli následující strukturu souborů a adresářů:

/ - kořenový adresář, kde se vyskytují skripty a pomocné soubory pro běh celého systému Tržiště

/admin/ – oddělená složka, ve které se nacházejí neveřejné soubory, které jsou využívány pro administrátorský subsystém. Přístup je chráněn administrátorským heslem.

/images/ - zde jsou umístěny grafické prvky, které používá celý systém

/includes/ - systémový adresář, ve kterém ve nacházejí hlavně konfigurační soubory, pomocí kterých se řeší např. připojení k databázi.

/menu/ - adresář s uživatelskou nabídkou

/templates/ - složka s HTML soubory formulářů pro PHP skripty

/uploaded/ - složka určená pro grafické soubory, které se ukládají jak pro potřeby administrátora, tak i uživatele.

3.7. Bezpečnost

Při realizaci tohoto projektu jsme pochopitelně museli vzít v úvahu i bezpečnost. Bylo třeba zajistit, aby do systému mohli zasahovat jen registrovaní uživatelé. Z toho důvodu má každý uživatel svoje uživatelské jméno a heslo, pod kterým se přihlašuje do systému.

Při ověřování totožnosti uživatelů jsme volili mezi dvěmi možnostmi. První možnost je taková, že uživatel se přihlásí pouze na začátku a pak již po celou dobu figuruje pod svým uživatelským jménem, takže již nemusí dále zadávat svoje uživatelské jméno ani heslo. Tento způsob je ale více náchylný k „nabourání“, protože systém je nucen při každém přechodu stránek ověřovat totožnost uživatele.

Proto jsme zvolili druhou možnost, při které může kdokoliv procházet Tržištěm a teprve při nějakém aktivním zásahu do systému, jako je např. přihození na aukci nebo vytvoření nové aukce, je vyžádáno uživatelské jméno a heslo. Důsledkem toho je sice povinnost uživatele opakovaně zadávat svoje přihlašovací údaje, např. když přihazuje na více aukcí, ale v praxi se tato situace vyskytne pouze výjimečně.

V několika málo případech (např. když uživatel prochází menu *Moje registrace*) jsme přesto byli nuceni kontrolovat totožnost uživatele při přechodu mezi stránkami. K tomuto účelu jsme použili technologii *sessions*, která ve stručnosti funguje tak, že po přihlášení uživatele se vygeneruje unikátní identifikační číslo, které je uloženo do databáze (entita *sessions*). Toto číslo je potom předáváno při přechodech mezi stránkami a tím je ověřována totožnost. Zároveň se kontroluje i doba platnosti takto vytvořeného čísla. Jestliže uživatel dlouho nepracuje s aplikací, platnost vyprší a je potřeba znovu zadat přihlašovací údaje.

Samostatnou kapitolou je potom zabezpečení přístupu administrátora. Zde se u přihlašování využívají *cookies*. Přístupové heslo administrátora, resp. administrátorů, je navíc v databázi uloženo zakódované technologií MD5. Tato funkce (technologie) vygeneruje pro libovolný řetězec 128bitové číslo, které je s velikou pravděpodobností jedinečné. V srpnu 2004 byly nelezeny nějaké kolize, díky nimž je tato technologie nepoužitelná v digitálních podpisech, ale jako základní bezpečnostní prvek pro uložení hesel v relačních databázích bohatě stačí.

Bylo také třeba ošetřit situace, kdy se zobrazuje nějaký vstup od uživatele. Pokud by se to provádělo bez jakékoliv kontroly, uživatel může do textu vložit nějaké HTML tagy, ale také povely JavaScriptu. Vložený HTML kód je mnohdy i žádoucí (vložení odkazu, zvýraznění textu,…), ale může zničit vzhled celé stránky. Neuzavřená tabulka způsobí, že se zbytek kódu již nezobrazí. Vložení meta-tagu pro refresh může způsobit přesměrování stránky úplně jinam. Vložený JavaScript může nejen poškodit vzhled stránky, přesměrovat stránku, ale může pomocí různých technik způsobit závažný bezpečnostní problém. Jde o tzv. session stealing a cross-site skripting a z toho plynoucí možný přístup třeba k cookies, které si stránka ukládá.

V závislosti na tom, jak moc bezpečná má aplikace být, je třeba vyřadit celé tagy nebo vyřadit klíčová slova jako onmouseout, onmouseover anebo překonvertovat znaky <> na entity < >. Tagy se tak překonvertují na běžný text a nebudou se interpretovat (toto například umožní vkládání ukázek HTML kódu). [5]

V PHP jsme pro předávaní textů použili tyto funkce:

$text = htmlspecialchars($text);

- speciální znaky budou převedeny na entity, HTML kód se nebude interpretovat a bude zobrazen jako plain-text

$text=ereg_replace("<[^>]+>", "", $text);

- regulární výraz, který odstraní tagy

$allowtags = <marquee>, <blink>, <hr>, <ul>, <li>, <ol>, <p>, <br>, <font>, <b>, <u>,<i>, <small>,<big>,,,<a>,<img>";

$text = strip_tags($text,$allowtags);

- odstranění nevhodných tagů v případě, kdy chceme alespoň nějaké tagy povolit a nezáleží nám až tak kriticky na bezpečnosti

4. Závěr

Diplomová práce úspěšně vyřešila komunikační problém mezi obchodujícími firmami v rámci Agrární komory v Liberci. Aplikace si kladla za cíl především usnadnit a zefektivnit obchodní jednání, tak aby jednotlivé firmy mohli snáze nabízet své produkty a poté je prodat tomu, kdo nabídne nejvyšší cenu.

Pro návrh a vývoj aplikace bylo nezbytným úkolem seznámit se důkladně s prostředím, kterého se práce týká – internetovými aukčními systémy. Teprve díky pečlivé analýze těchto systému bylo možné navrhnout vlastní systém, tak aby vyhovoval obecným pravidlům pro aukční obchodování a zamezilo se tak vývoji špatným směrem. Bylo také třeba nastudovat a následně implementovat internetové technologie Apache, MySQL, PHP, HTML a JavaScript.

Na konci vývojového cyklu uživatelské části systému lze konstatovat, že aplikace je plně funkční a splňuje zadavatelem kladené požadavky.

Při návrhu aplikace bylo rovněž snahou, aby v budoucnu nebyl problém rozšířit systém o další možné funkce. Předpokládá se např., že se časem přidá funkce „přihazující robot“, která umožní uživateli určit si nejvyšší přijatelnou cenu a aplikace bude až po tuto cenu přihazovat automaticky za něj.

Literatura:

[1] Klára Císařová. *Studijní materiály k předmětu RDB.* [online]. TU Liberec, 2003. <http://www.fm.vslib.cz/~ksi/cz/mater/rdb/Texty.zip>.

[2] Roman Špánek. *Návrh, vývoj a implementace vnitropodnikového informačního systému založeného na třívrstvé architektuře*. [Diplomová práce]. TU Liberec, 2003.

[3] Miluše Kutínová. *Zpracování hromadných dat na počítači.* VUT Brno, 2004.

[4] Jiří Hlavenka, Radek Sedlář, Tomáš Holčík, Jiří Kubeš, Jakub Mach. *Vytváříme WWW stránky a spravujeme moderní WEB SITE.* Praha, 1999.

[5] Martin Tobolka. *Třívrstvá aplikace pro párování nabídky a poptávky – administrátorská část systém, zadávání inzerátů a „černá listina“ nesolidních uživatelů.* [Diplomová práce]. TU Liberec, 2005

[6] Jiří Kosek. *PHP – Tvorba interaktivních internetových aplikací.* Praha, 1999.

[7] *Informační server zaměřený na vývoj e-commerce řešení*. [online]. Barcelona,2004. <http://phpauction.org>.

[8] *Informační server zabývající se informačními technologiemi* <http://interval.cz>

Příloha

Výpis SQL skriptu pro vygenerování relací v phpMyAdmin

```
-- phpMyAdmin SQL Dump
-- version 2.6.0-pl3
-- http://www.phpmyadmin.net
-- Verze MySQL: 4.0.24
-- Verze PHP: 4.3.9
-- Databáze: `trziste`

CREATE DATABASE `trziste`;
USE trziste;

-- Struktura tabulky `admin`

CREATE TABLE `admin` (
  `id` int(4) NOT NULL auto_increment,
  `email` varchar(50) NOT NULL default '',
  `password` varchar(30) NOT NULL default '',
  PRIMARY KEY  (`id`)
) TYPE=MyISAM AUTO_INCREMENT=2 ;


-- Vypisuji data pro tabulku `admin`

INSERT INTO `admin` VALUES (1, 'nejakyemail@email.cz', 'tajneheslo');

-- Struktura tabulky `auctions`

CREATE TABLE `auctions` (
  `id` varchar(30) NOT NULL default '',
  `user` varchar(30) default NULL,
  `title` tinytext,
  `date` timestamp(14) NOT NULL,
  `description` text,
  `pict_url` tinytext,
  `category` int(11) default NULL,
  `minimum_bid` double(16,4) default NULL,
  `duration` char(2) default NULL,
  `location` tinytext,
  `location_zip` varchar(6) default NULL,
  `shipping` char(1) default NULL,
  `payment` tinytext,
  `international` char(1) default NULL,
  `ends` timestamp(14) NOT NULL default '00000000000000',
  `current_bid` double(16,4) default NULL,
  `closed` char(1) default NULL,
  `photo_uploaded` char(1) default NULL,
  `quantity` int(11) default NULL,
  `suspended` int(1) default '0',
  PRIMARY KEY  (`id`),
  KEY `id` (`id`)
) TYPE=MyISAM;

-- Struktura tabulky `bids`

CREATE TABLE `bids` (
  `auction` varchar(30) default NULL,
  `bidder` varchar(30) default NULL,
  `bid` double(16,4) default NULL,
  `bidwhen` timestamp(14) NOT NULL,
  `quantity` int(11) default '0'
) TYPE=MyISAM;

-- Struktura tabulky `categories`

CREATE TABLE `categories` (
  `cat_id` int(4) NOT NULL auto_increment,
  `parent_id` int(4) default NULL,
```

```
  `cat_name` tinytext,
  `deleted` int(1) default NULL,
  `sub_counter` int(11) default NULL,
  `counter` int(11) default NULL,
  `cat_colour` tinytext NOT NULL,
  `cat_image` tinytext NOT NULL,
  PRIMARY KEY  (`cat_id`)
) TYPE=MyISAM AUTO_INCREMENT=65 ;

-- Struktura tabulky `categories_plain`

CREATE TABLE `categories_plain` (
  `id` int(11) NOT NULL auto_increment,
  `cat_id` int(11) default NULL,
  `cat_name` tinytext,
  PRIMARY KEY  (`id`)
) TYPE=MyISAM AUTO_INCREMENT=1 ;

-- Struktura tabulky `counters`

CREATE TABLE `counters` (
  `users` int(11) default '0',
  `auctions` int(11) default '0'
) TYPE=MyISAM;

-- Struktura tabulky `countries`

CREATE TABLE `countries` (
  `country_id` int(8) NOT NULL auto_increment,
  `domain` varchar(8) default NULL,
  `country` varchar(40) NOT NULL default '',
  `countries_long` varchar(80) default NULL,
  `ZIP` varchar(8) NOT NULL default '',
  `capital` varchar(80) default NULL,
  `logo` blob,
  PRIMARY KEY  (`country_id`),
  KEY `ZIP` (`ZIP`)
) TYPE=MyISAM AUTO_INCREMENT=197 ;

-- Vypisuji data pro tabulku `countries`

INSERT INTO `countries` VALUES (65, 'cz', 'Česko', 'Česká republika', 'CZ', 'Praha', NULL);
INSERT INTO `countries` VALUES (196, 'sk', 'Slovensko', 'Slovenská republika', 'SK', 'Bratislava',
NULL);

-- Struktura tabulky `durations`

CREATE TABLE `durations` (
  `days` int(2) NOT NULL default '0',
  `description` varchar(30) default NULL
) TYPE=MyISAM;

-- Vypisuji data pro tabulku `durations`

INSERT INTO `durations` VALUES (1, '1 den');
INSERT INTO `durations` VALUES (3, '3 dny');
INSERT INTO `durations` VALUES (7, '7 dní');
INSERT INTO `durations` VALUES (14, '14 dní');
INSERT INTO `durations` VALUES (30, '30 dní');
```

```
-- Struktura tabulky `feedbacks`

CREATE TABLE `feedbacks` (
  `rated_user_id` varchar(30) default NULL,
  `rater_user_nick` varchar(25) default NULL,
  `feedback` mediumtext,
  `rate` int(2) default NULL,
  `date` timestamp(14) NOT NULL
) TYPE=MyISAM;

-- Struktura tabulky `help`

CREATE TABLE `help` (
  `topic` varchar(40) default NULL,
  `helptext` text
) TYPE=MyISAM;

-- Vypisuji data pro tabulku `help`

INSERT INTO `help` VALUES ('Jak pou?ívat TR?I?TĚ', 'Intuitivně ... :-)');
INSERT INTO `help` VALUES ('Registrace', 'Přečtěte si pozorně podmínky registrace v přísluš?né
části');

-- Struktura tabulky `increments`

CREATE TABLE `increments` (
  `id` decimal(3,0) default NULL,
  `low` double(16,4) default NULL,
  `high` double(16,4) default NULL,
  `increment` double(16,4) default NULL
) TYPE=MyISAM;

-- Vypisuji data pro tabulku `increments`

INSERT INTO `increments` VALUES (1, 0.0000, 9.9900, 1.0000);
INSERT INTO `increments` VALUES (2, 10.0000, 99.9900, 5.0000);
INSERT INTO `increments` VALUES (3, 100.0000, 299.9900, 8.0000);
INSERT INTO `increments` VALUES (4, 300.0000, 599.9900, 10.0000);
INSERT INTO `increments` VALUES (5, 600.0000, 999.9900, 25.0000);
INSERT INTO `increments` VALUES (6, 1000.0000, 4999.9900, 50.0000);
INSERT INTO `increments` VALUES (7, 5000.0000, 29999.9900, 100.0000);

-- Struktura tabulky `news`

CREATE TABLE `news` (
  `topic` varchar(40) default NULL,
  `newstext` text
) TYPE=MyISAM;

-- Vypisuji data pro tabulku `news`

INSERT INTO `news` VALUES ('Proběhla úprava menu', 'Proběhla úprava menu');
INSERT INTO `news` VALUES ('Kategorie', 'Byly přidány barvy');
INSERT INTO `news` VALUES ('Ji? jsou funkční obrázky', 'Funguje FTP přenos souborů');
INSERT INTO `news` VALUES ('Změna hesel a e-mailu', 'Administrátor mů?e měnit administrátorské
heslo a e-mail');
```

```
-- Struktura tabulky `payments`

CREATE TABLE `payments` (
  `id` int(2) default NULL,
  `description` varchar(30) default NULL
) TYPE=MyISAM;


-- Vypisuji data pro tabulku `payments`

INSERT INTO `payments` VALUES (1, 'Bankovní převod');
INSERT INTO `payments` VALUES (2, 'Hotově');
INSERT INTO `payments` VALUES (3, 'Jinak dohodou');

-- Struktura tabulky `sessions`

CREATE TABLE `sessions` (
  `id` varchar(33) default NULL,
  `vars` text,
  `created` timestamp(14) NOT NULL,
  `last_visit` timestamp(14) NOT NULL default '00000000000000'
) TYPE=MyISAM;


-- Struktura tabulky `users`

CREATE TABLE `users` (
  `id` varchar(30) NOT NULL default '',
  `nick` varchar(25) default NULL,
  `password` varchar(20) default NULL,
  `name` tinytext,
  `address` tinytext,
  `city` varchar(25) default NULL,
  `prov` varchar(10) default NULL,
  `country` varchar(4) default NULL,
  `zip` varchar(6) default NULL,
  `phone` varchar(40) default NULL,
  `email` varchar(50) default NULL,
  `reg_date` timestamp(14) NOT NULL,
  `rate_sum` int(11) default NULL,
  `rate_num` int(11) default NULL,
  `birthdate` int(8) default NULL,
  `suspended` int(1) default '0',
  KEY `id` (`id`)
) TYPE=MyISAM;

-- Struktura tabulky `view_counter`

CREATE TABLE `view_counter` (
  `auctionsid` char(30) default NULL,
  `c_view` int(11) default '0'
) TYPE=MyISAM;
```